Volume II

Matematica delle Reti Neurali

Architetture avanzate · Sicurezza · Grafi spettrali · Ragionamento · Frontiere della ricerca

Silvano Salvador · Continuazione del Volume I (Parti 1–50)

PREFAZIONE

Introduzione al Volume II

Questo volume prosegue e approfondisce la trattazione del Volume I, esplorando le frontiere più avanzate e recenti del deep learning.

Il Volume I ha gettato le fondamenta: dalla definizione rigorosa di vettore e matrice fino all'analisi della convergenza degli ottimizzatori, passando per la backpropagation tensoriale, l'architettura Transformer completa, i modelli generativi, l'apprendimento per rinforzo e la statistica bayesiana. Il presente volume non presuppone altri materiali oltre a quello: ogni rimando al Volume I è esplicito e accompagnato da una breve ricapitolazione del concetto rilevante.

Il filo conduttore è la coerenza metodologica: ogni affermazione è fondata su una derivazione, ogni concetto è ancorato a una definizione formale, ogni architettura è smontata nei suoi componenti matematici elementari. Al tempo stesso si mantiene l'attenzione alle motivazioni pratiche — perché una certa scelta progettuale migliora le prestazioni, quale problema risolve, quale compromesso introduce.

Struttura del volume

Il testo è organizzato in otto parti tematiche:

- **Parte A (§51–53):** Teoria spettrale dei grafi, GNN spettrali, e modelli multimodali visione-linguaggio
- **Parte B (§54–55):** Adversarial examples, robustezza certificata, differential privacy, federated learning
- **Parte C (§56–60):** Architetture open-weight (Mistral, Mixtral, Gemma), Mamba-2, interpretabilità meccanicistica, sicurezza dei LLM
- **Parte D (§61–64):** Optimal Transport, rappresentazioni implicite (NeRF, SIREN), AlphaFold, foundation models e leggi di scala
- **Parte E (§65–67):** In-context learning, chain-of-thought, retrieval-augmented generation
- **Parte F (§68):** Modelli per il parlato: WaveNet, Tacotron, Whisper
- **Parte G (§69–72):** Neural Architecture Search, Hypernetworks, Capsule Networks, Spiking Neural Networks

- **Parte H (§73–75):** Quantum Machine Learning, Optimal Control, teoria del campionamento
- **Parte I (§76–78):** Score-Based SDE, reti equivarianti, Flow Matching
- **Parte L (§79–82):** Teoria della misura, landscape, MADE, VQ-VAE, sequenze lunghe (S4, Hyena)
- **Parte M (§83–87):** Domain adaptation, previsione probabilistica, loss avanzate, dati tabulari, position encoding
- **Parte N (§88–92):** Alignment (KTO, IPO, SimPO), Glow, Nesterov, NTK, memoria Hopfield
- **Parte O (§93–100):** GAN stabili, Information Bottleneck, Federated avanzato, Capsule, SNN, QML, Optimal Control, aliasing

> 📖 **Riferimento al Volume I:** i simboli e le notazioni seguono fedelmente la Sezione 0 del Volume I (Parte 0 — Notazioni e Prerequisiti). In caso di dubbio su un simbolo, si rimanda a quella sezione.

ABSTRACT

Il presente volume costituisce la continuazione autonoma del *Volume I — Matematica delle Reti Neurali* (Parti 0–50). Copre cinquanta sezioni (Parti 51–100) organizzate in quindici parti tematiche, trattando le frontiere più avanzate e recenti del deep learning con rigore matematico uniforme: ogni affermazione è fondata su una derivazione esplicita, ogni simbolo è definito nel glossario in Appendice A.

I contributi principali del volume includono: il framework SDE unificato per i modelli generativi score-based (VP-SDE, VE-SDE, Probability Flow ODE, Classifier-Free Guidance); la teoria dell'equivarianza di gruppo con G-CNN ed EGNN; il Flow Matching con trasporto ottimale; l'analisi del landscape non convesso con SAM e flat minima; il Neural Tangent Kernel e il regime di lazy training; le Modern Hopfield Networks e la loro equivalenza con il meccanismo di attenzione; gli State Space Models S4 e Hyena per sequenze ultra-lunghe; i metodi di alignment DPO, KTO, IPO e SimPO con derivazioni complete; i modelli autoregressivi MADE e VQ-VAE; la teoria della misura come fondamento dell'ERM; la domain adaptation con i bound di Ben-David; la conformal prediction con garanzie di copertura esatta; le Spiking Neural Networks con surrogate gradient e STDP; i Variational Quantum Circuits con parameter-shift rule e barren plateau; il Principio del Massimo di Pontryagin e l'equazione HJB con connessioni alle Neural ODE; la teoria del campionamento e l'aliasing nelle CNN.

Il volume si chiude con cinque appendici tecniche: glossario dei simboli (sette tabelle), identità matematiche di riferimento (algebra lineare, calcolo, SDE), complessità computazionale delle principali architetture e scaling laws, iperparametri consigliati e checklist di diagnostica del training, e una bibliografia commentata di oltre ottanta lavori originali e di riferimento.

PARTE 51

Teoria Spettrale dei Grafi

Il Laplaciano del grafo come operatore differenziale discreto, la sua decomposizione spettrale, e il passaggio alla convoluzione spettrale che fonda le GNN moderne.

51.1 Il Laplaciano del Grafo

Sia $G = (V, E)$ un grafo non orientato con $|V| = N$ nodi e matrice di adiacenza $\mathbf{A} \in \{0,1\}^{N \times N}$. La **matrice dei gradi** è la diagonale $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_N)$ con $d_i = \sum_j A_{ij}$.

Laplaciano non normalizzato

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \in \mathbb{R}^{N \times N}$$

Per un segnale $\mathbf{f} \in \mathbb{R}^N$ definito sui nodi, il prodotto $(\mathbf{L}\mathbf{f})_i$ misura la *discrepanza locale* del segnale:

$$(\mathbf{L}\mathbf{f})_i = d_i f_i - \sum_{j \sim i} f_j = \sum_{j \sim i} (f_i - f_j)$$

Questa è l'analogia discreta del laplaciano continuo $\Delta f = \nabla^2 f$: misura quanto il valore in un nodo differisce dalla media dei vicini. Il quadratico associato ha un'interpretazione elegante:

$$\mathbf{f}^T \mathbf{L} \mathbf{f} = \sum_{(i,j) \in E} (f_i - f_j)^2 \geq 0$$

Dimostrazione: $\mathbf{f}^T\mathbf{L}\mathbf{f} = \mathbf{f}^T\mathbf{D}\mathbf{f} - \mathbf{f}^T\mathbf{A}\mathbf{f} = \sum_i d_i f_i^2 - \sum_{ij} A_{ij} f_i f_j = \frac{1}{2}\sum_{ij} A_{ij}(f_i - f_j)^2$. Il laplaciano è pertanto **semidefinito positivo**: $\mathbf{L} \succeq \mathbf{0}$, con tutti gli autovalori $\lambda_k \geq 0$.

Laplaciano normalizzato simmetrico

$$\mathbf{L}_{\mathrm{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

Gli autovalori di $\mathbf{L}_{\text{sym}}$ giacciono in $[0, 2]$. Questa versione è simmetrica e preferita nelle applicazioni di deep learning perché non è sensibile alla scala dei gradi.

Laplaciano random-walk

$$\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$$

La matrice $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ è la matrice di transizione di un random-walk: $P_{ij} = A_{ij}/d_i$ è la probabilità di spostarsi dal nodo i al nodo j in un passo. Il laplaciano random-walk misura quanto il segnale differisce dalla media pesata dei vicini.

51.2 Decomposizione Spettrale e Connettività

Poiché $\mathbf{L}$ è simmetrica semidefinita positiva, ammette decomposizione spettrale:

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \qquad \mathbf{\Lambda} = \text{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1}), \quad 0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{N-1}$$

dove $\mathbf{U} = [\mathbf{u}_0 | \mathbf{u}_1 | \cdots | \mathbf{u}_{N-1}]$ è la matrice ortogonale degli autovettori ($\mathbf{U}^T\mathbf{U} = \mathbf{I}$).

Connettività del grafo e molteplicità di $\lambda_0 = 0$

TEOREMA — CONNETTIVITÀ E SPETTRO

La molteplicità dell'autovalore zero di $\mathbf{L}$ è uguale al numero di componenti connesse di G. In particolare:

- G è connesso $\iff$ $\lambda_0 = 0$ ha molteplicità 1 $\iff$ $\lambda_1 > 0$
- λ_1 (gap spettrale) misura la connettività: più è grande, più il grafo è difficile da partizionare

Dimostrazione del caso connesso: Se $\mathbf{L}\mathbf{f} = \mathbf{0}$, allora $\mathbf{f}^T\mathbf{L}\mathbf{f} = \sum_{(i,j)\in E}(f_i - f_j)^2 = 0$, quindi $f_i = f_j$ per ogni arco (i, j). In un grafo connesso, ogni nodo è raggiungibile da ogni altro tramite un cammino di archi, quindi $f_i = c$ (costante) per ogni i. Il nucleo di $\mathbf{L}$ è dunque monodimensionale, generato da $\mathbf{u}_0 = \mathbf{1}/\sqrt{N}$.

Cheeger's Inequality: connessione tra λ_1 e isoperimetria

Sia $h(G)$ il **rapporto di Cheeger** (o edge expansion): $h(G) = \min_{S \subset V,\, |S| \leq N/2} \frac{|E(S, \bar{S})|}{\min(|S|, |\bar{S}|)}$, dove $E(S, \bar{S})$ sono gli archi tra S e il suo complemento. Vale:

$$\frac{\lambda_1}{2} \leq h(G) \leq \sqrt{2\lambda_1}$$

Il bound superiore (Cheeger) mostra che un piccolo gap spettrale implica un taglio economico — utile per la *graph clustering spectral*.

51.3 Graph Fourier Transform

Analogamente alla Trasformata di Fourier discreta (DFT), che decompone un segnale nelle frequenze (autofunzioni del laplaciano continuo), la **Graph Fourier Transform** decompone un segnale $\mathbf{f} \in \mathbb{R}^N$ nelle frequenze del grafo (autovettori di $\mathbf{L}$):

$$\hat{\mathbf{f}} = \mathbf{U}^T \mathbf{f} \in \mathbb{R}^N \qquad \text{(coefficienti spettrali)}$$

$$\mathbf{f} = \mathbf{U} \hat{\mathbf{f}} \qquad \text{(ricostruzione)}$$

Gli autovettori $\mathbf{u}_k$ corrispondenti a λ_k piccoli (basse frequenze) variano lentamente tra nodi vicini — sono le componenti "smooth". Quelli con λ_k grandi (alte frequenze) variano rapidamente e catturano variazioni locali bruschi.

51.4 Convoluzione Spettrale su Grafi

La convoluzione tra due segnali $\mathbf{f}$ e $\mathbf{g}$ sul grafo si definisce come moltiplicazione punto per punto nel dominio spettrale (analogamente al teorema di convoluzione di Fourier):

$$(\mathbf{f} \star \mathbf{g}) = \mathbf{U}(\hat{\mathbf{f}} \odot \hat{\mathbf{g}}) = \mathbf{U}\, \mathrm{diag}(\hat{\mathbf{g}})\, \mathbf{U}^T \mathbf{f}$$

Un filtro spettrale g_θ parametrizzato da θ produce:

$$\mathbf{h} = g_\theta(\mathbf{L})\mathbf{f} = \mathbf{U}\, g_\theta(\mathbf{\Lambda})\, \mathbf{U}^T \mathbf{f}$$

dove $g_\theta(\mathbf{\Lambda}) = \mathrm{diag}(g_\theta(\lambda_0), \dots, g_\theta(\lambda_{N-1}))$. **Problemi:** (1) la decomposizione spettrale di $\mathbf{L}$ costa $O(N^3)$; (2) il filtro non è localizzato nello spazio — ciascun nodo di output dipende da *tutti* i nodi di input.

51.5 Approssimazione di Chebyshev (ChebNet)

Hammond et al. (2011) e Defferrard et al. (2016) approssimano $g_\theta(\lambda)$ con un polinomio di Chebyshev di grado K:

$$g_\theta(\lambda) \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\lambda}), \qquad \tilde{\lambda} = \frac{2\lambda}{\lambda_{\max}} - 1 \in [-1, 1]$$

dove T_k è il polinomio di Chebyshev di primo tipo, definito dalla ricorrenza:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_k(x) = 2x\,T_{k-1}(x) - T_{k-2}(x)$$

Il filtro diventa $g_\theta(\mathbf{L}) = \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{L}})$ con $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{\max} - \mathbf{I}$. La chiave: $T_k(\tilde{\mathbf{L}})$ può essere calcolato ricorsivamente *senza mai calcolare* $\mathbf{U}$, con complessità $O(K|E|)$. Il filtro ha supporto spaziale K-hop: ogni nodo di output dipende solo dal suo vicinato a K passi.

PARTE 52

GNN Spettrali: Derivazione Rigorosa della GCN e Oversmoothing

52.1 Derivazione Passo per Passo della GCN

Kipf e Welling (2017) semplificano ChebNet con $K = 1$ e $\lambda_{\max} \approx 2$, ottenendo $\tilde{\mathbf{L}} \approx \mathbf{L} - \mathbf{I}$:

1. Con $K = 1$: $g_\theta(\mathbf{L}) \approx \theta_0 \mathbf{I} + \theta_1(\mathbf{L} - \mathbf{I}) = \theta_0 \mathbf{I} + \theta_1(\mathbf{D}^{-1/2}\mathbf{L}_{\text{sym}}\mathbf{D}^{1/2} - \mathbf{I})$
   Approssimazione $\lambda_{\max} \approx 2$, $K = 1$ in ChebNet
2. Poniamo $\theta = \theta_0 = -\theta_1$ (un solo parametro): $g_\theta(\mathbf{L}) \approx \theta(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})$
   Sostituzione $\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ e raccolta
3. Il termine $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ ha autovalori in $[0, 2]$ — può causare instabilità. Si applica il **renormalization trick**: $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{A}_{ij}$
   Aggiunta di self-loop per normalizzare gli autovalori in $[0, 1]$
4. Risultato finale: $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, con autovalori in $[0, 1]$
   Normalizzazione simmetrica della matrice di adiacenza con self-loop

$$\mathbf{H}^{(\ell+1)} = \sigma\Big(\hat{\mathbf{A}}\,\mathbf{H}^{(\ell)}\,\mathbf{W}^{(\ell)}\Big), \qquad \hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$$

L'elemento (i, j) di $\hat{\mathbf{A}}$ è $\hat{A}_{ij} = \tilde{A}_{ij}/\sqrt{\tilde{d}_i \tilde{d}_j}$. L'aggiornamento del nodo i diventa:

$$\mathbf{h}_i^{(\ell+1)} = \sigma\left(\sum_{j \in \tilde{\mathcal{N}}(i)} \frac{1}{\sqrt{\tilde{d}_i \tilde{d}_j}}\, \mathbf{h}_j^{(\ell)}\, \mathbf{W}^{(\ell)}\right)$$

52.2 Oversmoothing: Analisi Matematica

Con L layer GCN, ogni nodo aggrega informazioni dal suo vicinato a L hop. Per $L \to \infty$, le rappresentazioni di tutti i nodi convergono alla stessa distribuzione stazionaria — fenomeno detto **oversmoothing**.

Formalmente, l'operazione di aggregazione è $\mathbf{H}^{(L)} = \hat{\mathbf{A}}^L \mathbf{H}^{(0)} \mathbf{W}$. Poiché $\hat{\mathbf{A}}$ è simmetrica con autovalori $\mu_i \in [0, 1]$:

$$\hat{\mathbf{A}}^L = \mathbf{U} \boldsymbol{\mu}^L \mathbf{U}^T \xrightarrow{L \to \infty} \mathbf{u}_0 \mathbf{u}_0^T = \frac{\mathbf{1}\mathbf{d}^T}{\sum_i d_i}$$

dove l'unico autovalore che sopravvive è $\mu_0 = 1$ (corrispondente all'autovettore stazionario del random-walk). Tutti i nodi convergono alla stessa rappresentazione pesata per il grado — perdita totale di informazione nodale distintiva. In pratica, già per $L \geq 8$ le prestazioni degradano significativamente su molti dataset.

Soluzioni all'oversmoothing

- **Residual connections:** $\mathbf{H}^{(\ell+1)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}) + \mathbf{H}^{(\ell)}$ — il termine residuale preserva le feature originali
- **Jumping Knowledge Networks:** $\mathbf{h}_i^{\text{final}} = f_{\text{agg}}(\mathbf{h}_i^{(1)}, \ldots, \mathbf{h}_i^{(L)})$ — si usa un aggregatore sull'output di tutti i layer
- **DropEdge:** rimozione casuale di archi durante il training — riduce il mixing spettrale
- **PairNorm:** normalizzazione che mantiene la distanza media tra nodi costante attraverso i layer

52.3 GraphSAGE: Inductive Learning

GCN è *transductive*: richiede l'intero grafo durante il training e non generalizza a nodi non visti. GraphSAGE (Hamilton et al., 2017) impara una funzione di aggregazione applicabile a qualsiasi nodo:

$$\mathbf{h}_{\mathcal{N}(v)}^{k} = \mathrm{AGG}_k\big(\{\mathbf{h}_u^{k-1} : u \in \mathcal{N}(v)\}\big)$$

$$\mathbf{h}_v^{k} = \sigma\Big(\mathbf{W}^k \left[\mathbf{h}_v^{k-1} \| \mathbf{h}_{\mathcal{N}(v)}^{k}\right]\Big)$$

La concatenazione del nodo stesso con l'aggregazione del vicinato preserva l'identità nodale. Funzioni di aggregazione supportate:

Aggregatore	Formula	Proprietà
Mean	$\frac{1}{\lvert\mathcal{N}(v)\rvert}\sum_{u\in\mathcal{N}(v)}\mathbf{h}_u$	GCN approssimato
Max pooling	$\max_{u\in\mathcal{N}(v)}\sigma(\mathbf{W}_{\text{pool}}\mathbf{h}_u+\mathbf{b})$	Cattura feature estreme
LSTM	LSTM su permutazione casuale di $\mathcal{N}(v)$	Più espressivo, non-invariante

GraphSAGE usa **neighborhood sampling**: campiona S_k vicini al layer k, rendendo il training scalabile a grafi con milioni di nodi — complessità $O(\prod_k S_k)$ per nodo invece di $O(N)$.

PARTE 53

Modelli Multimodali: Visione e Linguaggio

CLIP, Flamingo, LLaVA e la matematica della fusione cross-modale.

53.1 CLIP: Contrastive Language-Image Pre-Training

CLIP (Radford et al., 2021) impara rappresentazioni congiunte di immagini e testi tramite apprendimento contrastivo su N coppie (I_i, T_i) (immagine, didascalia). Due encoder separati producono embedding normalizzati:

$$\mathbf{v}_i = f_V(I_i)/\|f_V(I_i)\|, \qquad \mathbf{t}_i = f_T(T_i)/\|f_T(T_i)\|$$

La matrice di similarità coseno $\mathbf{S} \in \mathbb{R}^{N\times N}$ ha $S_{ij} = \mathbf{v}_i^T \mathbf{t}_j \cdot \exp(\tau)$ dove τ è la temperatura (apprendibile). La **InfoNCE loss** (bidirectional) è:

$$\mathcal{L}_{\text{CLIP}} = -\frac{1}{2N}\sum_{i=1}^{N}\left[\log\frac{e^{S_{ii}}}{\sum_j e^{S_{ij}}} + \log\frac{e^{S_{ii}}}{\sum_j e^{S_{ji}}}\right]$$

Il primo termine ottimizza la riga i (immagine i verso tutti i testi), il secondo la colonna i (testo i verso tutte le immagini). Con batch size N, ci sono N coppie positive e $N(N-1)$ coppie negative.

Gradiente rispetto alla temperatura τ

La temperatura scala la distribuzione softmax. Derivando la loss rispetto a τ (trattando $S_{ij} = \mathbf{v}_i^T\mathbf{t}_j$ come costante):

$$\frac{\partial \mathcal{L}}{\partial \tau} = -\frac{1}{2N}\sum_i\left[S_{ii} - \sum_j S_{ij}\,p_{ij}\right] - \frac{1}{2N}\sum_i\left[S_{ii} - \sum_j S_{ji}\,p_{ji}\right]$$

dove $p_{ij} = e^{S_{ij}}/\sum_k e^{S_{ik}}$ sono i pesi softmax. τ apprende automaticamente la "temperatura" ottimale per distinguere coppie positive da negative.

Zero-shot inference

Il vantaggio chiave di CLIP: dati i template testuali $\{\text{"una foto di un [classe]"}\}$ per K classi, la predizione zero-shot è:

$$\hat{y} = \arg\max_k \mathbf{v}_i^T \mathbf{t}_k$$

Nessun fine-tuning richiesto — il modello generalizza a qualsiasi classificazione descrivibile testualmente.

53.2 Flamingo: Cross-Attention Visione → Testo

Flamingo (Alayrac et al., 2022) integra informazioni visive in un LLM pre-addestrato tramite **gated cross-attention** inserita tra i layer del modello linguistico.

Perceiver Resampler

Le feature visive di un'immagine o video sono compressi in un numero fisso N_q di "visual token" tramite il Perceiver Resampler:

$$\mathbf{X}_{\text{vis}} = \text{CrossAttn}(\mathbf{Q}_{\text{learn}}, \mathbf{K}_V, \mathbf{V}_V) \in \mathbb{R}^{N_q \times d}$$

dove $\mathbf{Q}_{\text{learn}} \in \mathbb{R}^{N_q \times d}$ sono N_q query apprendibili (latent queries), e $\mathbf{K}_V, \mathbf{V}_V$ sono le proiezioni delle feature visive dense (es. $H \times W$ patch da un ViT). $N_q \ll H \cdot W$ — il resampler comprime l'informazione visiva.

Gated Cross-Attention Layer

Tra ogni coppia di layer del LLM, Flamingo inserisce un layer di cross-attention gated:

$$\mathbf{y} = \mathbf{x} + \tanh(\alpha) \cdot \text{CrossAttn}(\text{LN}(\mathbf{x}), \mathbf{X}_{\text{vis}})$$
$$\mathbf{x}' = \mathbf{y} + \tanh(\beta) \cdot \text{FFN}(\text{LN}(\mathbf{y}))$$

I gate α, β (scalari apprendibili, inizializzati a 0) permettono al modello di ignorare l'informazione visiva all'inizio del training — il LLM pre-addestrato rimane invariato e la visione viene integrata gradualmente.

53.3 LLaVA: Visual Instruction Tuning

LLaVA (Liu et al., 2023) usa un approccio più semplice: un **proiettore lineare** (o MLP) che mappa le feature visive di CLIP nello spazio degli embedding del LLM:

$$\mathbf{H}_v = \mathbf{W} \cdot f_V(I) + \mathbf{b} \in \mathbb{R}^{N_p \times d_{\text{LLM}}}$$

I token visivi $\mathbf{H}_v$ vengono concatenati con i token testuali e processati dal LLM come una sequenza unificata. La semplicità è il punto di forza: nessuna architettura cross-attention aggiuntiva — il LLM apprende a "leggere" i token visivi come se fossero token testuali.

Training in due fasi

Fase	Cosa si addestra	Dati	Obiettivo
Pre-training	Solo il proiettore $\mathbf{W}$	595K coppie immagine-didascalia	Allineare feature visive allo spazio LLM
Fine-tuning	Proiettore + LLM	158K istruzioni multimodali	Seguire istruzioni che coinvolgono immagini

53.4 Metriche di Valutazione Multimodale

La valutazione dei modelli multimodali usa metriche specializzate. Il **CIDEr score** (Consensus-based Image Description Evaluation) misura la similarità delle didascalie generate con le reference tramite TF-IDF su n-grammi:

$$\text{CIDEr}_n(c, S) = \frac{1}{|S|} \sum_{s_i \in S} \frac{\mathbf{g}^n(c) \cdot \mathbf{g}^n(s_i)}{\|\mathbf{g}^n(c)\| \|\mathbf{g}^n(s_i)\|}$$

dove $\mathbf{g}^n(c)$ è il vettore TF-IDF degli n-grammi della caption c. Il punteggio finale è $\text{CIDEr} = \frac{1}{N} \sum_n w_n \, \text{CIDEr}_n$.

PARTE 54

Adversarial Examples e Robustezza Certificata

La matematica degli attacchi avversariali, le difese certificate, e il training avversariale come problema min-max.

54.1 FGSM: Derivazione dal Gradiente del Segno

Un **esempio avversariale** è un input $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ impercettibilmente diverso dall'originale ($\|\boldsymbol{\delta}\|_\infty \leq \varepsilon$) che inganna il classificatore: $f(\tilde{\mathbf{x}}) \neq f(\mathbf{x})$. Il **Fast Gradient Sign Method** (FGSM, Goodfellow et al., 2015) costruisce la perturbazione ottimale per la norma L_∞:

$$\boldsymbol{\delta}_{\text{FGSM}} = \varepsilon \cdot \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y))$$

Motivazione: La norma L_∞ della perturbazione è $\|\boldsymbol{\delta}\|_\infty = \varepsilon$. Per massimizzare la loss con vincolo $\|\boldsymbol{\delta}\|_\infty \leq \varepsilon$, si vuole massimizzare $\boldsymbol{\delta}^T \nabla_{\mathbf{x}} \mathcal{L}$ soggetto a $\|\boldsymbol{\delta}\|_\infty \leq \varepsilon$. La soluzione è $\delta_i = \varepsilon \cdot \text{sign}(\partial \mathcal{L} / \partial x_i)$ — ogni componente prende il valore estremo nella direzione del gradiente.

ESEMPIO NUMERICO — FGSM

Sia $\nabla_{\mathbf{x}} \mathcal{L} = (0.3, -0.1, 0.7, -0.4)^T$ e $\varepsilon = 0.02$:

$$\boldsymbol{\delta}_{\text{FGSM}} = 0.02 \cdot (+1, -1, +1, -1)^T = (0.02, -0.02, 0.02, -0.02)^T$$

La perturbazione è dello stesso ordine di grandezza dell'errore di quantizzazione delle immagini (1/255 ≈ 0.004), ma in direzione ottimale per massimizzare la loss.

54.2 PGD: Attacco Iterativo con Proiezione

PGD (Projected Gradient Descent, Madry et al., 2018) è l'attacco FGSM iterato con passo α e proiezione sulla ε-ball:

$$\boldsymbol{\delta}^{(0)} \sim \mathcal{U}([-\varepsilon, \varepsilon]^d) \qquad \text{(inizializzazione casuale)}$$
$$\boldsymbol{\delta}^{(t+1)} = \Pi_\varepsilon\Big(\boldsymbol{\delta}^{(t)} + \alpha \cdot \operatorname{sign}(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x} + \boldsymbol{\delta}^{(t)}, y))\Big)$$

dove $\Pi_\varepsilon(\boldsymbol{\delta}) = \operatorname{clip}(\boldsymbol{\delta}, -\varepsilon, \varepsilon)$ è la proiezione sulla ℓ_∞-ball di raggio ε. PGD con molti passi trova il massimo locale della loss — è considerato il più forte attacco a scatola bianca (white-box) pratico.

Attacco L_2

Per la norma L_2, la proiezione è diversa: $\Pi_\varepsilon^{L_2}(\boldsymbol{\delta}) = \varepsilon\, \boldsymbol{\delta} / \max(\|\boldsymbol{\delta}\|_2, \varepsilon)$. L'aggiornamento usa il gradiente normalizzato:

$$\boldsymbol{\delta}^{(t+1)} = \Pi_\varepsilon^{L_2}\left(\boldsymbol{\delta}^{(t)} + \alpha \cdot \frac{\nabla_{\mathbf{x}} \mathcal{L}}{\|\nabla_{\mathbf{x}} \mathcal{L}\|_2}\right)$$

54.3 Robustezza Certificata: Randomized Smoothing

La robustezza certificata garantisce formalmente che $f(\tilde{\mathbf{x}}) = f(\mathbf{x})$ per ogni $\|\boldsymbol{\delta}\|_2 \le r$ — non solo per gli attacchi noti. Il **Randomized Smoothing** (Cohen et al., 2019) costruisce un classificatore *smoothed*:

$$g(\mathbf{x}) = \arg\max_c \; P_{\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})}[f(\mathbf{x} + \boldsymbol{\varepsilon}) = c]$$

TEOREMA (COHEN ET AL., 2019)

Sia $p_A = P[f(\mathbf{x} + \boldsymbol{\varepsilon}) = c_A]$ la probabilità della classe più probabile. Se $p_A \ge \underline{p_A} > \frac{1}{2}$, allora $g(\mathbf{x} + \boldsymbol{\delta}) = c_A$ per ogni $\|\boldsymbol{\delta}\|_2 \le r$ con:

$$r = \frac{\sigma}{2}\Big(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(1 - \underline{p_A})\Big) = \sigma\, \Phi^{-1}(\underline{p_A})$$

dove Φ^{-1} è la funzione quantile della gaussiana standard.

Più confidenza nella classe predetta ($\underline{p_A}$ vicino a 1) → certificato di raggio più grande. Il rumore σ controlla il trade-off: σ grande → certezza su perturbazioni maggiori, ma accuratezza su dati puliti più bassa.

54.4 Adversarial Training: Formulazione Min-Max

L'adversarial training (Madry et al., 2018) risolve il problema min-max:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)} \left[\max_{\|\boldsymbol{\delta}\|_p \leq \varepsilon} \mathcal{L}(f_\theta(\mathbf{x}+\boldsymbol{\delta}), y) \right]$$

Il problema esterno (min su θ) è il training standard; il problema interno (max su $\boldsymbol{\delta}$) è la ricerca dell'attacco più forte. In pratica si alterna: (1) inner maximization con PGD; (2) outer minimization con un passo SGD sull'esempio avversariale trovato.

Teorema di convergenza: per funzioni di loss fortemente convesse in θ e concave in $\boldsymbol{\delta}$, il metodo converge al punto di sella $(\theta^*, \boldsymbol{\delta}^*)$ con tasso $O(1/T)$. In pratica la loss non è né convessa né concava, ma l'alternanza empiricamente converge.

PARTE 55

Privacy nel Machine Learning

Differential privacy, DP-SGD, federated learning e membership inference: la matematica della protezione dei dati sensibili.

55.1 Differential Privacy: Definizione Formale

DEFINIZIONE — (ε, δ)-DIFFERENTIAL PRIVACY

Un meccanismo randomizzato $\mathcal{M} : \mathcal{D} \to \mathcal{O}$ è (ε,δ)-differenzialmente privato se per ogni coppia di dataset adiacenti D, D' (differenti in un solo elemento) e per ogni sottoinsieme di output $S \subseteq \mathcal{O}$:

$$P[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot P[\mathcal{M}(D') \in S] + \delta$$

Il parametro ε (privacy budget) misura la perdita di privacy: $\varepsilon = 0$ è perfetta privacy, ε grande permette inferenza sull'appartenenza. δ è la probabilità che la garanzia e^{ε} sia violata.

Meccanismo Gaussiano

Per una funzione $f : \mathcal{D} \to \mathbb{R}^k$ con **sensibilità** L_2 $\Delta_2 f = \max_{D,D'} \|f(D) - f(D')\|_2$, il meccanismo gaussiano è:

$$\mathcal{M}(D) = f(D) + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}), \qquad \sigma \geq \frac{\Delta_2 f \sqrt{2 \ln(1.25/\delta)}}{\varepsilon}$$

Questo garantisce (ε,δ)-DP. La varianza del rumore cresce con la sensibilità e decresce con ε — protezione maggiore richiede più rumore.

55.2 DP-SGD: Training Differenzialmente Privato

DP-SGD (Abadi et al., 2016) rende l'SGD differenzialmente privato con due modifiche: gradient clipping e aggiunta di rumore gaussiano.

ALGORITMO DP-SGD

Per ogni step t con mini-batch $\mathcal{B}$ di size L:

1. Per ogni esempio $i \in \mathcal{B}$, calcola e clippa il gradiente:

$$\tilde{\mathbf{g}}_{t,i} = \nabla_\theta \mathcal{L}(f_\theta(\mathbf{x}_i), y_i) \cdot \min\left(1, \frac{C}{\|\nabla_\theta \mathcal{L}_i\|_2}\right)$$

2. Aggiungi rumore gaussiano al gradiente medio:

$$\tilde{\mathbf{g}}_t = \frac{1}{L}\left(\sum_{i \in \mathcal{B}} \tilde{\mathbf{g}}_{t,i} + \mathcal{N}(\mathbf{0}, \sigma^2 C^2 \mathbf{I})\right)$$

3. Aggiorna i parametri: $\theta_{t+1} = \theta_t - \alpha \tilde{\mathbf{g}}_t$

Il clipping limita la sensibilità L_2 di ogni gradiente a C (clipping threshold). L'aggiunta di rumore σC nasconde il contributo di ogni singolo esempio. Il privacy budget consumato dopo T step si calcola con il **Moments Accountant** (analisi più stretta del composition theorem):

$$\varepsilon(T) = O\left(\frac{q\sqrt{T\log(1/\delta)}}{\sigma}\right), \qquad q = L/N \text{ (subsampling rate)}$$

55.3 Federated Learning: FedAvg

Il federated learning addestra un modello globale su dati distribuiti tra K client (dispositivi), senza mai centralizzare i dati. L'obiettivo:

$$\min_\theta \mathcal{L}(\theta) = \sum_{k=1}^{K} \frac{n_k}{N} \mathcal{L}_k(\theta), \qquad N = \sum_k n_k$$

L'algoritmo **FedAvg** (McMahan et al., 2017) alterna aggregazione globale e aggiornamenti locali:

$$\theta_k^{(t+1)} = \theta^{(t)} - \alpha \nabla \mathcal{L}_k(\theta^{(t)}) \quad (E \text{ passi SGD locali})$$

$$\theta^{(t+1)} = \sum_{k=1}^{K} \frac{n_k}{N} \theta_k^{(t+1)} \quad (\text{aggregazione pesata})$$

Il problema chiave: la **heterogeneità dei dati** (dati non i.i.d. tra client) causa *client drift* — ogni client converge verso il proprio minimo locale invece del minimo globale. La divergenza cumulativa dopo E step locali è:

$$\mathbb{E}\|\theta_k^{(t+E)} - \theta^*\|^2 \leq O\big(E^2 \alpha^2 \|\nabla \mathcal{L}_k(\theta^*) - \nabla \mathcal{L}(\theta^*)\|^2\big)$$

FedProx aggiunge un termine di prossimità per limitare il drift: $\mathcal{L}_k^{\text{FedProx}} = \mathcal{L}_k + \frac{\mu}{2}\|\theta - \theta^{(t)}\|^2$.

55.4 Membership Inference Attack

Un **membership inference attack** (Shokri et al., 2017) tenta di determinare se un esempio $(\mathbf{x}, y)$ era nel training set di un modello f_θ. L'attacco si basa sull'osservazione che i modelli tendono a essere più confidenti sugli esempi di training (overfitting):

$$\hat{m}(\mathbf{x}, y) = \mathbf{1}[P_\theta(y|\mathbf{x}) \geq \tau] \quad (\text{attacco basato su threshold})$$

Attacchi più sofisticati addestrano un modello shadow $f_{\theta_{\text{shadow}}}$ sugli stessi dati di distribuzione, e usano le sue predizioni per calibrare la threshold. L'**advantage** dell'attacco misura la violazione di privacy:

$$\text{Adv} = |P[\hat{m} = 1 | \text{membro}] - P[\hat{m} = 1 | \text{non-membro}]|$$

Un modello DP-addestrato con ε piccolo ha $\text{Adv} \leq e^\varepsilon - 1 \approx \varepsilon$ per $\varepsilon \ll 1$ — la DP fornisce garanzie teoriche sull'advantage massimo di qualsiasi attacco.

PARTE 56

Architettura Mistral 7B e Mixtral 8×7B

Sliding Window Attention, rolling buffer KV-cache, e la matematica interna del primo modello denso e del primo MoE open-weight di riferimento.

56.1 Sliding Window Attention

L'attenzione standard ha complessità $O(n^2)$ per sequenze di lunghezza n. Mistral (Jiang et al., 2023) usa la **Sliding Window Attention (SWA)**: ogni token attende solo ai W token precedenti (finestra scorrevole di ampiezza W):

$$A_{ij} = \begin{cases} \text{softmax}(q_i^T k_j / \sqrt{d_k}) & \text{se } i - W \leq j \leq i \\ 0 & \text{altrimenti} \end{cases}$$

Complessità: $O(nW)$ invece di $O(n^2)$. Per $W = 4096$ e $n = 32768$: $8\times$ di riduzione. Nonostante ogni layer abbia finestra W, l'informazione si propaga attraverso i layer: al layer L, un token può ricevere informazioni da token distanti fino a $L \cdot W$ posizioni.

Rolling Buffer KV-Cache

Con SWA, la KV-cache non deve crescere indefinitamente — si mantiene solo un buffer circolare di dimensione W. Alla posizione i, la K e V del token i vengono scritte nello slot $i \mod W$:

$$\text{Cache}[i \mod W] \leftarrow (k_i, v_i)$$

La memoria KV-cache è costante in n: $O(W \cdot d_k \cdot L)$ invece di $O(n \cdot d_k \cdot L)$. Per sequenze molto lunghe questo è il risparmio dominante.

56.2 Grouped Query Attention in Mistral

Mistral usa GQA con $n_h = 32$ teste query e $n_{kv} = 8$ teste key-value (4 query per gruppo). Il risparmio in KV-cache rispetto a MHA standard:

$$\text{Memoria KV (MHA)} = 2 \cdot n_h \cdot d_k \cdot n \quad \rightarrow \quad \text{Memoria KV (GQA)} = 2 \cdot n_{kv} \cdot d_k \cdot n$$
$$\text{Riduzione} = n_h / n_{kv} = 32/8 = 4\times$$

Per la quality del modello, GQA è equivalente a MHA su scala: Ainslie et al. (2023) mostrano che con il giusto fine-tuning da un checkpoint MHA, GQA recupera quasi interamente le prestazioni.

56.3 Mixtral 8×7B: MoE a Rete Densa Equivalente

Mixtral (Jiang et al., 2024) sostituisce ogni FFN con 8 esperti $\{E_i\}_{i=1}^{8}$, attivandone solo $K = 2$ per token. Il router è una proiezione lineare:

$$G(\mathbf{x}) = \text{TopK}(\text{softmax}(\mathbf{x}\mathbf{W}_g), K)$$
$$y = \sum_{i \in \text{TopK}(G(\mathbf{x}))} G_i(\mathbf{x}) \cdot E_i(\mathbf{x})$$

I pesi del router sono normalizzati solo sui K esperti selezionati: $\sum_{i \in \text{TopK}} G_i = 1$. La rete ha $8 \times 7\text{B} = 56\text{B}$ parametri totali, ma usa solo $2/8 = 25\%$ dei parametri FFN per token → complessità computazionale equivalente a un modello da ~13B.

Load balancing loss in Mixtral

Il router tende al collasso verso pochi esperti. La auxiliary loss:

$$\mathcal{L}_{\text{aux}} = \alpha \cdot N_e \cdot \sum_{i=1}^{N_e} f_i \cdot P_i$$
$$f_i = \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}[\text{token } t \to \text{esperto } i], \quad P_i = \frac{1}{T} \sum_{t=1}^{T} \frac{e^{(\mathbf{x}_t \mathbf{W}_g)_i}}{\sum_j e^{(\mathbf{x}_t \mathbf{W}_g)_j}}$$

Minimizzare il prodotto $f_i \cdot P_i$ scoraggia sia l'invio frequente di token all'esperto i (alto f_i) sia l'assegnazione di peso elevato all'esperto i (alto P_i).

PARTE 57

Architettura Gemma, PaLM e Varianti Moderne

57.1 Gemma: Scelte Architetturali

Gemma (Google DeepMind, 2024) è una famiglia di modelli open-weight che incorpora le migliori pratiche architetturali emerse dopo LLaMA. Le scelte principali:

Multi-Query Attention (MQA)

Gemma 2B usa MQA ($n_{kv} = 1$, un solo gruppo di K,V condiviso tra tutte le teste query). Risparmio massimo in KV-cache: $n_h\times$. La perdita di qualità rispetto a MHA per modelli <7B è trascurabile se il modello è sufficientemente addestrato.

GeGLU come attivazione FFN

Gemma usa GeGLU in tutti i layer FFN:

$$\text{FFN}_{\text{GeGLU}}(\mathbf{x}) = \text{GELU}(\mathbf{x}\mathbf{W}_1) \odot (\mathbf{x}\mathbf{V}) \cdot \mathbf{W}_2$$

La derivata rispetto a $\mathbf{x}$ coinvolge sia la derivata di GELU che quella lineare:

$$\frac{\partial}{\partial \mathbf{x}}\text{FFN}_{\text{GeGLU}} = \left[\text{GELU}'(\mathbf{x}\mathbf{W}_1) \odot (\mathbf{x}\mathbf{V})\right]\mathbf{W}_1^T\mathbf{W}_2^T + \left[\text{GELU}(\mathbf{x}\mathbf{W}_1)\right]\mathbf{V}^T\mathbf{W}_2^T$$

RoPE con base θ aumentata

Gemma usa RoPE con base $\theta_{\text{base}} = 10000$ come il Transformer originale, ma PaLM 2 e versioni successive usano $\theta_{\text{base}} = 10000^{10/9} \approx 15265$ — questo estende la lunghezza di contesto efficace senza ri-addestrare, sfruttando la proprietà di interpolazione di RoPE.

57.2 PaLM: Architettura Parallela

PaLM (Chowdhery et al., 2022) introduce un'innovazione architetturale: **parallel layers**, in cui il sublayer di attenzione e la FFN vengono calcolati *in parallelo* anziché in sequenza:

$$\mathbf{y} = \mathbf{x} + \mathrm{MultiHead}(\mathrm{LN}(\mathbf{x})) + \mathrm{FFN}(\mathrm{LN}(\mathbf{x}))$$

Invece della formula standard sequenziale $\mathbf{x}' = \mathbf{x} + \mathrm{MH}(\mathrm{LN}(\mathbf{x}))$, $\mathbf{y} = \mathbf{x}' + \mathrm{FFN}(\mathrm{LN}(\mathbf{x}'))$. Il vantaggio computazionale: le proiezioni $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$ e $\mathbf{W}_1^{\mathrm{FFN}}$ possono essere fuse in un'unica grande moltiplicazione matrice-matrice, migliorando l'utilizzo dell'hardware del 15% tipicamente.

57.3 Estensione del Contesto con RoPE: YaRN

YaRN (Peng et al., 2023) estende la lunghezza di contesto di modelli RoPE senza ri-addestrare da zero. L'idea: per dimensioni con alta frequenza ω_i (grande), interpola le posizioni; per bassa frequenza (piccola), estrapolazione diretta.

$$\tilde{\theta}_i = \begin{cases} \omega_i & \text{se } \lambda_i < 1 \text{ (dimensioni ad alta freq.)} \\ (1 - s^{-1})\omega_i \cdot L_{\mathrm{orig}}/(2\pi) + s^{-1}\omega_i & \text{se } \lambda_i \geq 1 \end{cases}$$

dove $\lambda_i = 2\pi/\omega_i$ è la lunghezza d'onda della dimensione i, L_{orig} è la lunghezza di training originale, e $s = L_{\mathrm{new}}/L_{\mathrm{orig}}$ è il fattore di scala. Il fine-tuning su una piccola quantità di dati lunghi completa l'adattamento.

PARTE 58

Mamba-2 e State Space Duality (SSD)

La connessione profonda tra SSM e attenzione lineare, e il framework SSD che unifica entrambe le famiglie.

58.1 State Space Duality

Dao e Gu (2024) dimostrano che l'SSM selettivo di Mamba è matematicamente equivalente a una forma di **attenzione con matrice strutturata**. Il punto di partenza: la ricorrenza SSM

$$h_t = A_t h_{t-1} + B_t x_t, \qquad y_t = C_t h_t$$

può essere "srotolata" in forma di sequenza. Definiamo $M_{ts} = C_t \left(\prod_{j=s+1}^{t} A_j\right) B_s$ per $t \geq s$. Allora:

$$y_t = \sum_{s=1}^{t} M_{ts}\, x_s = (\mathbf{M}\mathbf{x})_t, \qquad \mathbf{M} \in \mathbb{R}^{T \times T} \text{ triangolare inferiore}$$

Questo è un'**attenzione con matrice semi-separabile**: la matrice $\mathbf{M}$ ha rango strutturato — ogni sotto-matrice rettangolare estratta dalla parte triangolare inferiore ha rango al più N (la dimensione dello stato). Con $A_t = a_t \mathbf{I}$ (scalare), la struttura diventa:

$$M_{ts} = a_{t:s}\, \mathbf{c}_t^T \mathbf{b}_s, \qquad a_{t:s} = \prod_{j=s+1}^{t} a_j$$

Questa è esattamente l'attenzione lineare con kernel $k(\mathbf{q}_t, \mathbf{k}_s) = a_{t:s}\, \mathbf{q}_t^T \mathbf{k}_s$, dove il fattore di decadimento $a_{t:s}$ dipende dalla distanza temporale.

58.2 Algoritmo Chunk-wise per Mamba-2

La dualità SSM $\leftrightarrow$ Attenzione suggerisce un algoritmo efficiente che combina i vantaggi di entrambe le modalità di computazione. Dividiamo la sequenza in chunk di dimensione Q:

$$\underbrace{\text{Intra-chunk}}_{\text{come attenzione, parallelizzabile}} : \quad y_t = \sum_{s \in \text{chunk}(t)} M_{ts} x_s$$

$$\underbrace{\text{Inter-chunk}}_{\text{come SSM, ricorrente}} : \quad h_{\text{end-of-chunk}} = \prod_{j \in \text{chunk}} A_j \cdot h_{\text{prev}} + \sum_{j \in \text{chunk}} (\cdot)$$

Complessità: $O(T \cdot Q \cdot d)$ per la parte intra-chunk + $O(T/Q \cdot N \cdot d)$ per la parte inter-chunk. Scegliendo $Q = \sqrt{T}$: $O(T^{3/2} d)$ — un compromesso tra $O(T^2 d)$ (attenzione full) e $O(TNd)$ (SSM puro).

58.3 Confronto Analitico SSM vs Attenzione

Proprietà	Attenzione (Softmax)	SSM (Mamba)
Complessità training	$O(T^2 d)$	$O(TNd)$ con scan parallelo
Complessità inferenza (per token)	$O(Td)$ (KV-cache cresce)	$O(Nd)$ (stato costante)
Memoria inferenza	$O(T)$ (KV-cache lineare)	$O(N)$ (stato fisso)
Content-based retrieval	Esatto (softmax è selettivo)	Approssimato (decay su informazione vecchia)
Contesto infinito	No (KV-cache cresce)	Sì (stato compresso fisso)
Inductive bias	Nessuno (full attention)	Decadimento esponenziale nel tempo

PARTE 59

Interpretabilità Meccanicistica Avanzata

Superposition hypothesis, sparse autoencoders, logit lens e la scoperta di circuiti computazionali nei Transformer.

59.1 Superposition Hypothesis

Le reti neurali hanno tipicamente molte più feature semantiche da rappresentare che neuroni disponibili. La **superposition hypothesis** (Elhage et al., 2022) afferma che le reti risolvono questo problema codificando multiple feature in singoli neuroni tramite interferenza quasi-ortogonale.

Formalmente: si ha superposizione quando le attivazioni di un layer possono essere scritte come:

$$\mathbf{h} = \mathbf{W}\mathbf{f} + \boldsymbol{\eta}$$

dove $\mathbf{f} \in \mathbb{R}^m$ sono le feature con $m \gg d$ (molte più feature che neuroni), $\mathbf{W} \in \mathbb{R}^{d \times m}$ è la matrice di embedding, e $\boldsymbol{\eta}$ è il rumore di interferenza. Affinché le feature siano recuperabili, le colonne di $\mathbf{W}$ devono essere quasi-ortogonali. Per un grafo aléatorio in $\mathbb{R}^d$, il numero massimo di vettori quasi-ortogonali (con interferenza $\leq \varepsilon$) è $O(e^{cd})$ per una costante $c > 0$ — esponenzialmente più di d.

Trade-off sparsità-superposizione

Se le feature sono sparse (attive raramente), la probabilità di collisione è bassa e la superposizione è tollerabile. Con sparsità s (probabilità di attivazione), l'interferenza media su una feature i è:

$$\mathbb{E}[\text{interferenza su } i] = s \sum_{j \neq i} (\mathbf{w}_i^T \mathbf{w}_j)^2 \leq \frac{s(m-d)}{d}$$

Per s piccolo (feature sparse), la rete può codificare $m \gg d$ feature con interferenza controllata.

59.2 Sparse Autoencoders per Feature Discovery

Per identificare le feature in superposizione, si addestra uno **sparse autoencoder (SAE)** sull'output di un layer. Il SAE impara una base sovracompleta $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{m \times d}$ ($m \gg d$) con sparsità indotta dalla L1:

$$\mathbf{f} = \text{ReLU}(\mathbf{W}_{\text{enc}}(\mathbf{h} - \mathbf{b}_{\text{pre}}) + \mathbf{b}_{\text{enc}})$$

$$\hat{\mathbf{h}} = \mathbf{W}_{\text{dec}}\mathbf{f} + \mathbf{b}_{\text{dec}}$$

$$\mathcal{L}_{\text{SAE}} = \underbrace{\|\mathbf{h} - \hat{\mathbf{h}}\|^2}_{\text{ricostruzione}} + \lambda \underbrace{\|\mathbf{f}\|_1}_{\text{sparsità}}$$

Ogni colonna di $\mathbf{W}_{\text{dec}}$ corrisponde a una feature: se il dizionario è abbastanza grande (m tra $\times 4$ e $\times 64$ rispetto a d), ciascuna feature apprende a rispondere a concetti semantici interpretabili (città, generi musicali, stati emotivi, ecc.).

59.3 Logit Lens e Tuned Lens

Il **logit lens** (nostalgebraist, 2020) analizza cosa predice la rete a ogni layer intermedio. Applicando direttamente l'unembedding $\mathbf{W}_U$ all'output di ogni layer ℓ:

$$\hat{y}^{(\ell)} = \text{softmax}(\mathbf{h}^{(\ell)}\mathbf{W}_U)$$

Questo rivela come la predizione si "raffina" attraverso i layer: i layer iniziali catturano token frequenti, quelli mediani sviluppano struttura sintattica, quelli finali risolvono ambiguità semantiche.

Il **tuned lens** (Belrose et al., 2023) corregge il bias introdotto dall'applicazione prematura dell'unembedding addestrando un proiettore affine $T_\ell : \mathbf{h}^{(\ell)} \mapsto \tilde{\mathbf{h}}^{(\ell)}$ che minimizza la distanza KL con la predizione finale:

$$T_\ell^* = \arg\min_{T} \mathbb{E}\Big[D_{\text{KL}}(\hat{y}^{(L)} \| T(\mathbf{h}^{(\ell)})\mathbf{W}_U)\Big]$$

59.4 Circuiti: Il Task IOI

L'**Indirect Object Identification (IOI)** è il task di riferimento per circuit discovery. Data la frase "Quando Mary e John andarono in negozio, John diede a ___", il modello deve completare con "Mary".

Wang et al. (2022) identificano il circuito responsabile usando activation patching su tutte le teste di attenzione e i layer MLP. Il circuito comprende tre tipi di teste:

Tipo di testa	Funzione	Layer tipici
Name Mover Heads	Copiano il nome del soggetto nella posizione di output	Layer 9-11
Inhibition Heads	Sopprimono il nome duplicato (non-IO)	Layer 7-8
S-Inhibition Heads	Identificano quale nome è il soggetto ripetuto	Layer 5-6

Il circuito è identificato tramite l'algoritmo di activation patching: si misura $\Delta B_{\text{patch}}(c) = B(\text{patch}(\mathbf{x}_c, \mathbf{x}_p, c)) - B(\mathbf{x}_p)$ per ogni componente c, dove B è la differenza di logit tra IO e S (soggetto). I componenti con ΔB_{patch} alto sono causalmente necessari.

PARTE 60

Sicurezza dei Modelli Linguistici

Attacchi adversariali ai LLM, jailbreaking gradiente-based, Constitutional AI, scalable oversight e la matematica del reward hacking.

60.1 GCG Attack: Jailbreaking Gradiente-Based

Il **Greedy Coordinate Gradient (GCG)** (Zou et al., 2023) è il primo attacco automatico efficace ai LLM allineati. L'obiettivo: trovare un suffix avversariale $\mathbf{s}$ da concatenare al prompt $\mathbf{p}$ tale da massimizzare la probabilità che il modello produca una risposta dannosa.

Formalmente, si minimizza la loss negativa sul target $\mathbf{t}$ (es. "Certo, ecco come..."):

$$\min_{\mathbf{s}} \mathcal{L}(\mathbf{p} \| \mathbf{s}, \mathbf{t}) = -\log P_\theta(\mathbf{t} \mid \mathbf{p} \| \mathbf{s})$$

Il problema: $\mathbf{s}$ è una sequenza di token discreti — non differenziabile. GCG usa la linearizzazione del gradiente per stimare il token migliore in ogni posizione i:

$$\nabla_{\mathbf{e}_{s_i}} \mathcal{L} \in \mathbb{R}^{|V|} \qquad \text{(gradiente rispetto all'embedding one-hot del token } i\text{)}$$

$$s_i^* = \arg\min_{v \in \mathcal{V}} \left[\nabla_{\mathbf{e}_{s_i}} \mathcal{L} \right]_v$$

Ad ogni iterazione si campionano k candidati con sostituzione greedy, si valuta la loss su tutti, e si prende il migliore. GCG è transferable: suffix trovati su modelli open-source spesso funzionano su modelli closed-source.

60.2 Constitutional AI e RLAIF

Constitutional AI (Bai et al., 2022) propone di usare un LLM come supervisore di se stesso. Data una "Costituzione" (lista di principi come "sii utile, innocuo, onesto"), il training avviene in due fasi:

Fase SL-CAI (Supervised Learning):

1. Il modello genera una risposta potenzialmente dannosa
2. Lo stesso modello la critica rispetto a un principio della Costituzione
3. Il modello revisa la risposta incorporando la critica
4. Si addestra su $(prompt, revisione)$

Fase RL-CAI (RLAIF): invece di usare feedback umani, si usa il modello SL-CAI come reward model — **RL from AI Feedback**. Per coppie di risposte (y_A, y_B) al prompt x:

$$r(x, y_A, y_B) = P_{\text{CAI}}(\text{"A è migliore di B secondo il principio [P]"} \mid x, y_A, y_B)$$

Questo reward model viene poi usato in PPO come nel RLHF standard. Il vantaggio: scalabilità (nessun human labeler), consistenza (il modello AI applica i principi uniformemente).

60.3 Scalable Oversight e Debate

Il problema del **scalable oversight**: come verificare le risposte di un modello superintelligente quando le domande sono troppo difficili per gli umani? Due approcci teorici:

Amplification (Christiano et al., 2018)

Si costruisce un supervisore amplificato $\text{Amp}(H)$ ricorsivamente: H risponde a domande che può risolvere in tempo T, e $\text{Amp}(H)$ risponde usando n copie di H in tempo nT. Se H può supervisionare $\text{Amp}(H)$ (perché le sottoproblemi sono più semplici), si itera per ottenere supervisori di potenza arbitraria.

AI Debate (Irving et al., 2018)

Due agenti AI dibattono la risposta corretta davanti a un giudice umano. Claim: se il dibattito è a **zero somma**, l'agente onesto ha una strategia vincente indipendentemente dalla complessità della risposta — il giudice deve solo valutare il dibattito, non la risposta direttamente.

Formalizzazione come gioco estensivo: sia $V(s, a_1, a_2)$ il valore del gioco nello stato s con argomenti a_1 (onesto) e a_2 (disonesto). All'equilibrio di Nash:

$$V^* = \max_{a_1} \min_{a_2} J(s, a_1, a_2) \qquad \text{(onesto massimizza, disonesto minimizza)}$$

60.4 Goodhart's Law e Reward Hacking: Analisi Formale

La legge di Goodhart afferma: "Quando una misura diventa un obiettivo, cessa di essere una buona misura." Nel contesto dei LLM, questo si manifesta come **reward hacking**: il modello ottimizza $r_\phi(x, y)$ trovando comportamenti con reward alto che non corrispondono all'intento umano.

Krakovna et al. (2020) formalizzano il reward hacking tramite la **KL penalty**. Sia R^* il reward vero (intento umano) e $\hat{R}$ il proxy (reward model). La loss RLHF è:

$$\mathcal{L}_{\text{RLHF}} = \mathbb{E}[\hat{R}(y|x)] - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$$

La policy ottimale del proxy è $\pi^*_{\hat{R}}$; quella del reward vero è $\pi^*_{R^*}$. Il **KL-divergence scaling** mostra che:

$$R^*(\pi^*_{\hat{R}}) - R^*(\pi^*_{R^*}) \leq \sqrt{\frac{D_{\text{KL}}(\pi^*_{\hat{R}} \| \pi^*_{R^*})}{2} \cdot \text{Var}_{R^*}}$$

Il gap di performance reale è limitato dalla distanza KL tra le due policy ottimali — se $\hat{R}$ è un buon proxy, le policy sono vicine e il hacking è limitato. Aumentare β riduce il gap ma sacrifica la performance sul proxy.

Goodhart's Taxonomy

Tipo	Descrizione	Esempio in LLM
Regressional Goodhart	Proxy imperfetto del vero obiettivo	Reward model non cattura tutte le sfumature umane
Extremal Goodhart	Ottimizzare troppo il proxy porta fuori dalla distribuzione in cui vale la correlazione	Risposte molto lunghe ricevono reward alto ma sono meno utili
Causal Goodhart	Intervento sul proxy rompe la relazione causale	Il modello impara a sembrare utile senza esserlo
Adversarial Goodhart	Un agente manipola il proxy intenzionalmente	Il modello impara a fare sycophancy (dire quello che l'umano vuole sentire)

PARTE 61

Optimal Transport e Distanza di Wasserstein

La teoria del trasporto ottimale come fondamento matematico per confrontare distribuzioni di probabilità, con applicazioni a GAN, domain adaptation e generazione di immagini.

61.1 Il Problema di Monge-Kantorovich

Date due distribuzioni di probabilità μ e ν su $\mathbb{R}^d$, il **problema di trasporto ottimale di Monge** chiede di trovare una mappa $T : \mathbb{R}^d \to \mathbb{R}^d$ che trasporti μ in ν minimizzando il costo totale:

$$\inf_{T:\, T_\#\mu = \nu} \int_{\mathbb{R}^d} c(\mathbf{x}, T(\mathbf{x}))\, d\mu(\mathbf{x})$$

dove $c(\mathbf{x}, \mathbf{y})$ è il costo di trasportare una unità di massa da $\mathbf{x}$ a $\mathbf{y}$, e $T_\#\mu = \nu$ indica che T è una mappa di trasporto da μ a ν (push-forward). Il problema di Monge è *non convesso* — può non avere soluzione.

Rilassamento di Kantorovich

Kantorovich propose una versione rilassata: invece di una mappa deterministica, si cerca un **piano di trasporto** (coupling) $\gamma \in \Pi(\mu, \nu)$ — una distribuzione congiunta su $\mathbb{R}^d \times \mathbb{R}^d$ con marginali μ e ν:

$$\mathcal{OT}(\mu, \nu) = \inf_{\gamma \in \Pi(\mu,\nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} c(\mathbf{x}, \mathbf{y})\, d\gamma(\mathbf{x}, \mathbf{y})$$

Questo è un problema di ottimizzazione *lineare* (programmazione lineare in dimensione infinita). Per distribuzioni discrete con n punti ciascuna, si riduce a un problema di trasporto su una matrice $n \times n$, risolvibile in $O(n^3 \log n)$.

61.2 Distanza di Wasserstein

Con costo $c(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^p$, l'ottimale di Kantorovich definisce la **distanza di Wasserstein di ordine p**:

DEFINIZIONE — PWASSERSTEIN DISTANCE

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Pi(\mu,\nu)} \int \|\mathbf{x} - \mathbf{y}\|^p \, d\gamma(\mathbf{x}, \mathbf{y}) \right)^{1/p}$$

W_p è una vera metrica sullo spazio delle distribuzioni di probabilità con momento p-esimo finito. La metrica W_2 ha proprietà geometriche particolarmente ricche — è collegata alla geometria riemanniana dello spazio delle distribuzioni.

Dualità di Kantorovich-Rubinstein (W_1)

Per $p = 1$, la dualità forte fornisce una formulazione alternativa cruciale per le applicazioni:

TEOREMA DI KANTOROVICH-RUBINSTEIN

$$W_1(\mu, \nu) = \sup_{\|f\|_L \leq 1} \left[\int f \, d\mu - \int f \, d\nu \right] = \sup_{\|f\|_L \leq 1} \mathbb{E}_\mu[f] - \mathbb{E}_\nu[f]$$

dove il sup è su tutte le funzioni $f : \mathbb{R}^d \to \mathbb{R}$ con costante di Lipschitz $\|f\|_L = \sup_{\mathbf{x} \neq \mathbf{y}} |f(\mathbf{x}) - f(\mathbf{y})| / \|\mathbf{x} - \mathbf{y}\| \leq 1$.

Intuizione: $W_1(\mu, \nu)$ misura quanto si deve "spostare" la massa — la funzione di Lipschitz ottimale f^* è quella che "vede" la differenza tra μ e ν nel modo più informativo possibile.

61.3 WGAN: GAN con Distanza di Wasserstein

Arjovsky et al. (2017) sostituiscono la loss GAN standard (Jensen-Shannon divergence) con la W_1-distance, ottenendo training più stabile. Il critico (discriminatore) deve essere una funzione 1-Lipschitz:

$$\mathcal{L}_{\text{WGAN}} = \max_{\|f_w\|_L \leq 1} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[f_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[f_w(G_\theta(\mathbf{z}))]$$

Il vincolo Lipschitz viene implementato con **weight clipping** (WGAN originale: $w_i \in [-c, c]$) o con **gradient penalty** (WGAN-GP):

$$\text{Missing argument for \mathbf}$$

dove Missing argument for \mathbf con $t \sim \mathcal{U}[0,1]$ — punti interpolati tra dati reali e generati. La penalità impone $\|\nabla f_w\|_2 = 1$ lungo le linee rette tra distribuzione reale e generata, approssimando il vincolo Lipschitz.

Vantaggi rispetto alla GAN standard

La GAN standard minimizza $D_{\text{KL}}(p_g \| p_{\text{data}})$ che non è definita quando le due distribuzioni hanno supporti disgiunti (caso tipico nelle prime fasi del training). La W_1-distance è invece sempre ben definita e fornisce un gradiente informativo anche quando p_g e p_{data} non si sovrappongono.

61.4 Sinkhorn: Optimal Transport Entropico

Il problema OT discreto è $O(n^3)$ — troppo lento per dati ad alta dimensione. La regolarizzazione entropica introduce un termine di entropia:

$$\mathcal{OT}_\varepsilon(\mathbf{a}, \mathbf{b}) = \min_{\mathbf{P} \in U(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle - \varepsilon H(\mathbf{P})$$

dove $\mathbf{C}_{ij} = \|\mathbf{x}_i - \mathbf{y}_j\|^2$, $H(\mathbf{P}) = -\sum_{ij} P_{ij} \log P_{ij}$ e
$U(\mathbf{a}, \mathbf{b}) = \{\mathbf{P} \geq 0 : \mathbf{P}\mathbf{1} = \mathbf{a}, \mathbf{P}^T\mathbf{1} = \mathbf{b}\}$. La soluzione ha forma:

$$P^*_{ij} = u_i \, \exp(-C_{ij}/\varepsilon) \, v_j$$

con vettori $\mathbf{u}$ e $\mathbf{v}$ trovati dall'**algoritmo di Sinkhorn** — alternanza di normalizzazioni per riga e per colonna, convergenza in $O(n^2/\varepsilon^2)$, parallelizzabile su GPU.

PARTE 62

Implicit Neural Representations: NeRF e SIREN

Rappresentare segnali continui (scene 3D, immagini, audio) come funzioni parametrizzate da reti neurali, invece di griglie discrete.

62.1 NeRF: Neural Radiance Fields

NeRF (Mildenhall et al., 2020) rappresenta una scena 3D come una funzione $F_\theta : (\mathbf{x}, \mathbf{d}) \to (\mathbf{c}, \sigma)$ che mappa posizione $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ e direzione di visione $\mathbf{d} \in \mathbb{S}^2$ in colore $\mathbf{c} = (r, g, b) \in [0,1]^3$ e densità volumetrica $\sigma \geq 0$.

Volume Rendering

Il colore atteso di un raggio $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ (da origine $\mathbf{o}$ nella direzione $\mathbf{d}$) è dato dall'integrale di rendering volumetrico:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma(\mathbf{r}(t))\, \mathbf{c}(\mathbf{r}(t), \mathbf{d})\, dt$$

dove $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))\, ds\right)$ è la **trasmittanza accumulata** — la probabilità che il raggio non venga assorbito prima del punto t. Discretizzando su N campioni $t_1<\cdots

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \left(1 - e^{-\sigma_i \delta_i}\right) \mathbf{c}_i, \quad T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \quad \delta_i = t_{i+1} - t_i$$

Positional Encoding per NeRF

Le reti neurali apprendono con difficoltà variazioni ad alta frequenza se alimentate con coordinate dirette. NeRF usa positional encoding sinusoidale per proiettare le coordinate in uno spazio ad alta dimensione:

$$\gamma(p) = \left(\sin(2^0 \pi p), \cos(2^0 \pi p), \ldots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)\right)$$

Con $L = 10$ per le posizioni e $L = 4$ per le direzioni. Questo è analogo a una base di Fourier — permette alla rete di approssimare funzioni ad alta frequenza con un numero ragionevole di parametri.

Training: fotometric loss

$$\mathcal{L}_{\text{NeRF}} = \sum_{\mathbf{r} \in \mathcal{R}} \| \hat{C}(\mathbf{r}) - C_{\text{gt}}(\mathbf{r}) \|_2^2$$

$\mathcal{R}$ è il batch di raggi campionati dalle immagini di training. Nessuna supervisione 3D — solo fotografie da diverse angolazioni.

62.2 SIREN: Reti con Attivazioni Sinusoidali

SIREN (Implicit Neural Representations with Periodic Activation Functions, Sitzmann et al., 2020) usa $\sin$ come funzione di attivazione invece di ReLU:

$$\phi_i(\mathbf{x}) = \sin(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i)$$

La derivata di un layer SIREN è essa stessa un SIREN: $\partial \phi_i / \partial \mathbf{x}_i = \cos(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i) \cdot \mathbf{W}_i$. Questo rende SIREN adatto a rappresentare funzioni con informazione nelle derivate (es. risoluzione di PDE, signal fitting).

Inizializzazione speciale per SIREN

La distribuzione dell'input di ogni layer deve rimanere uniforme in $[-\pi, \pi]$ per mantenere la potenza espressiva. Sitzmann et al. derivano:

$$W_{ij} \sim \mathcal{U}\left[-\sqrt{6/n}, \sqrt{6/n}\right] \quad \text{(primo layer)}$$

$$W_{ij} \sim \mathcal{U}\left[-\sqrt{6/n} \cdot c, \sqrt{6/n} \cdot c\right] \quad \text{(layer successivi, } c = \sqrt{6}\text{)}$$

Si può dimostrare che con questa inizializzazione, la distribuzione di input di ogni layer rimane approssimativamente $\mathcal{U}[-1, 1]$, preservando la ricchezza delle rappresentazioni attraverso la profondità.

62.3 Instant-NGP: Hash Encoding

NeRF richiede ore di training. Müller et al. (2022) propongono la **multiresolution hash encoding**: invece di calcolare il positional encoding analiticamente, si usano tabelle di hash a risoluzioni multiple.

Per livello ℓ con risoluzione N_ℓ, le coordinate $\mathbf{x}$ vengono mappate nella griglia più vicina e i corner features vengono interpolati trilinearmente:

$$\mathbf{e}_\ell(\mathbf{x}) = \text{interp}\big(\mathbf{x},\, \{T_\ell[\text{hash}(\lfloor \mathbf{x} \cdot N_\ell \rfloor + \mathbf{o})]\}_{\mathbf{o}\in\{0,1\}^3}\big)$$

dove T_ℓ è la tabella di hash del livello ℓ e $\text{hash}(\mathbf{k}) = (k_1 \oplus k_2\pi_2 \oplus k_3\pi_3) \mod T$ con π_2, π_3 numeri primi grandi. Le collisioni di hash (due posizioni diverse mappate allo stesso entry) causano un rumore trascurabile in pratica. Il training è $100\times$ più veloce di NeRF originale.

PARTE 63

AlphaFold: Predizione della Struttura Proteica

Come DeepMind ha risolto un problema cinquantennale della biologia strutturale usando attenzione evolutiva e geometria 3D equivariante.

63.1 Multiple Sequence Alignment e Attenzione Evolutiva

Una proteina è una sequenza di amminoacidi $s_1, \ldots, s_L$ con $s_i \in \{\text{20 amminoacidi}\}$. La struttura 3D è determinata dalla sequenza, ma la mappatura è estremamente complessa.

AlphaFold2 (Jumper et al., 2021) sfrutta l'informazione evolutiva: per ogni proteina target si costruisce un **Multiple Sequence Alignment (MSA)** — centinaia/migliaia di sequenze omologhe di organismi diversi. La co-variazione tra posizioni i e j nell'MSA indica che quei residui sono in contatto nella struttura 3D.

Evoformer Block

L'Evoformer processa in parallelo due rappresentazioni:

- $\mathbf{M} \in \mathbb{R}^{N_{\text{seq}} \times L \times c_M}$ — rappresentazione MSA (N_{seq} sequenze, L residui)
- $\mathbf{Z} \in \mathbb{R}^{L \times L \times c_Z}$ — rappresentazione a coppie (relazioni tra residui i, j)

Le due rappresentazioni si aggiornano reciprocamente tramite **outer product mean**:

$$Z_{ij} \leftarrow Z_{ij} + \text{Linear}\left(\frac{1}{N_{\text{seq}}} \sum_{s=1}^{N_{\text{seq}}} \mathbf{m}_{si} \otimes \mathbf{m}_{sj} \right)$$

Il prodotto esterno $\mathbf{m}_{si} \otimes \mathbf{m}_{sj} \in \mathbb{R}^{c_M^2}$ cattura la co-variazione evolutiva tra posizioni i e j — se gli amminoacidi in quelle posizioni co-evolvono, i vettori $\mathbf{m}_{si}$ e $\mathbf{m}_{sj}$ diventeranno correlati attraverso le sequenze.

Triangle Attention

Per le rappresentazioni a coppie Z_{ij}, AlphaFold usa l'**attenzione triangolare** che sfrutta la disuguaglianza triangolare: se i residui i e k sono vicini, e k e j sono vicini, allora i e j probabilmente sono vicini:

$$Z_{ij} \leftarrow Z_{ij} + \sum_{k} \alpha_{ijk} \, g_{ijk} \, Z_{kj}$$

$$\alpha_{ijk} = \text{softmax}_k\left(\frac{1}{\sqrt{c}} (Z_{ij}\mathbf{W}^Q)(Z_{ik}\mathbf{W}^K)^T + b_{ij} \right)$$

Il bias $b_{ij} = Z_{ij}\mathbf{W}^B$ aggiunge informazione sulla relazione diretta i-j ai punteggi di attenzione.

63.2 Structure Module: Equivarianza Roto-Traslazionale

Il Structure Module converte le rappresentazioni dell'Evoformer in coordinate 3D. Ogni residuo i ha un frame di riferimento locale $T_i = (R_i, \mathbf{t}_i) \in SE(3)$ (gruppo speciale euclideo: rotazione + traslazione).

L'aggiornamento degli angoli diedri $(\phi_i, \psi_i, \omega_i, \boldsymbol{\chi}_i)$ (che determinano la conformazione locale della backbone) avviene tramite Invariant Point Attention (IPA):

$$a_{ij} = \frac{1}{\sqrt{c}} \left[(\mathbf{q}_i^T \mathbf{k}_j) + \sum_{h} w_h \left(-\frac{1}{2} \| T_i \mathbf{q}_i^h - T_j \mathbf{k}_j^h \|^2 \right) \right]$$

Il termine $\| T_i \mathbf{q}_i^h - T_j \mathbf{k}_j^h \|^2$ confronta punti nello spazio 3D tenendo conto dei frame locali — l'attenzione diventa equivariante rispetto a rotazioni e traslazioni globali della struttura.

PARTE 64

Foundation Models e Leggi di Scala

Quando e perché i modelli più grandi funzionano meglio: derivazione delle Chinchilla scaling laws, emergenza e transizioni di fase, e le implicazioni per il design dei modelli.

64.1 Chinchilla Scaling Laws: Derivazione

Hoffmann et al. (2022) modellano la loss di un LLM come funzione dei parametri N e dei token di training D:

$$\hat{\mathcal{L}}(N, D) = E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}$$

dove E è l'entropia irriducibile dei dati, A/N^α è il termine di parametri (la rete è troppo piccola), B/D^β è il termine di dati (non abbastanza training). I valori stimati: $\alpha \approx 0.34$, $\beta \approx 0.28$, $A \approx 406.4$, $B \approx 410.7$.

Allocazione ottimale del budget computazionale

Per un budget C FLOPs (con $C \approx 6ND$ per LLM), si vuole minimizzare $\hat{\mathcal{L}}$ soggetto a $6ND = C$:

1. Sostituiamo $D = C/(6N)$ nella loss: $\hat{\mathcal{L}} = E + A/N^\alpha + B(6N/C)^\beta$
   Sostituzione del vincolo
2. Deriviamo rispetto a N e poniamo uguale a zero:
   $-\alpha A/N^{\alpha+1} + \beta B(6/C)^\beta N^{\beta-1} = 0$
   Condizione del primo ordine
3. $N^{\alpha+\beta} = \frac{\alpha A}{\beta B}(C/6)^\beta$
   Raccogliendo N da entrambi i lati
4. $N^*(C) \propto C^{\beta/(\alpha+\beta)}, \quad D^*(C) \propto C^{\alpha/(\alpha+\beta)}$
   Sostituendo con $D = C/(6N)$

$$N^* \propto C^{0.45}, \qquad D^* \propto C^{0.55}, \qquad \frac{D^*}{N^*} \approx 20\text{–}30 \text{ token/parametro}$$

La raccomandazione pratica di Chinchilla: per ogni parametro del modello occorrono ~20 token di training. GPT-3 (175B parametri, 300B token) era *undertraining* — il modello ottimale per lo stesso budget computazionale è molto più piccolo ma più addestrato.

64.2 Emergenza e Transizioni di Fase

L'**emergenza** nei LLM si riferisce a capacità che appaiono improvvisamente al superamento di una soglia di scala — non c'è progressione graduale visibile in metriche standard.

Modello teorico: task con soglia

Wei et al. (2022) propongono che l'emergenza è legata a task che richiedono *tutti* i passi corretti di un algoritmo. Se ogni passo ha accuratezza p, e servono k passi, l'accuratezza totale è p^k. Questa funzione ha derivata quasi nulla per p piccolo e un salto brusco vicino a $p = 1$:

$$\frac{d}{dp}(p^k) = kp^{k-1} \xrightarrow{p \to 1} k$$

Per task complicati (k grande), la transizione da bassa ad alta accuratezza appare come una discontinuità al crescere della scala del modello.

Calibrazione delle metriche

Schaeffer et al. (2023) argomentano che l'emergenza scompare con metriche continue (es. log-loss o probabilità invece di accuratezza discreta) — la discontinuità è un artefatto della metrica, non del modello. La scelta della metrica è dunque fondamentale per interpretare correttamente l'emergenza.

PARTE 65

In-Context Learning: Teoria e Meccanismi

65.1 ICL come Inferenza Bayesiana Implicita

Xie et al. (2022) mostrano che l'ICL può essere interpretato come inferenza bayesiana. Dati k esempi in-context $\{(\mathbf{x}_i, y_i)\}_{i=1}^{k}$ e una query $\mathbf{x}_{k+1}$, un modello bayesiano ideale calcola:

$$P(y_{k+1} | \mathbf{x}_{k+1}, \mathbf{x}_{1:k}, y_{1:k}) = \int P(y_{k+1} | \mathbf{x}_{k+1}, \theta)\, p(\theta | \mathbf{x}_{1:k}, y_{1:k})\, d\theta$$

Il modello linguistico impara durante il preaddestramento la prior $p(\theta)$ sulle funzioni θ presenti nei testi, e il conditioning sul contesto implementa implicitamente l'aggiornamento bayesiano della posterior $p(\theta | \mathcal{D}_{\text{ctx}})$.

65.2 ICL come Gradient Descent Implicito

Akyürek et al. (2022) e Von Oswald et al. (2023) dimostrano che un layer Transformer con attenzione lineare implementa esattamente un passo di gradient descent sul contesto.

Sia il contesto $\{(\mathbf{x}_i, y_i)\}_{i=1}^{k}$ con $y_i = \mathbf{w}^* \cdot \mathbf{x}_i$. Un layer di attenzione lineare con $\mathbf{W}^Q = \mathbf{W}^K = \mathbf{I}$ e $\mathbf{W}^V$ appropriata computa:

$$\hat{\mathbf{w}} = \left(\sum_{i=1}^{k} \mathbf{x}_i \mathbf{x}_i^T \right)^{-1} \sum_{i=1}^{k} y_i \mathbf{x}_i = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

che è esattamente la soluzione di minimi quadrati — il gradiente descente converso del problema di regressione lineare sul contesto. Questo suggerisce che i Transformer sviluppano "ottimizzatori interni" durante il preaddestramento.

Aumento del numero di esempi in-context

Sia $\mathcal{L}_k$ la loss con k esempi in-context. Si ha la seguente relazione di scala (empirica):

$$\mathcal{L}_k \approx \mathcal{L}_0 - c \cdot \log(k+1)$$

La performance migliora logaritmicamente con il numero di esempi — il modello "impara" dal contesto in modo sub-lineare, analogamente alla curva di apprendimento umana.

PARTE 66

Chain-of-Thought e Ragionamento

66.1 La Matematica del Ragionamento Step-by-Step

Il **Chain-of-Thought prompting** (Wei et al., 2022) chiede al modello di produrre passi di ragionamento intermedi prima della risposta finale. Perché funziona?

Formalmente: per un task che richiede k passaggi di ragionamento, un LLM autoregressivo che produce direttamente la risposta deve rappresentare l'intera computazione in un singolo forward pass di dimensione d. Un LLM con CoT distribuisce la computazione in k token aggiuntivi, ognuno dei quali può usare l'intera larghezza del modello.

TEOREMA (FENG ET AL., 2023)

Esistono famiglie di problemi che richiedono profondità computazionale $T(n)$ e che:

- Non possono essere risolti da un Transformer di profondità L senza CoT (se $L < T(n)$)
- Possono essere risolti con CoT di lunghezza $O(T(n)/L)$

La CoT estende effettivamente la "profondità computazionale" del modello.

66.2 Ragionamento come Programma: Scratchpad Formale

Nye et al. (2021) propongono di addestrare modelli a usare uno **scratchpad** (area di lavoro esplicita). La loss diventa:

$$\mathcal{L} = -\log P_\theta(\text{scratchpad}|\text{input}) - \lambda \log P_\theta(\text{risposta}|\text{input}, \text{scratchpad})$$

Il modello impara a produrre computazioni intermedie esplicite, trasferendo il carico computazionale dal "peso dei parametri" alla "lunghezza del contesto".

Process Reward vs Outcome Reward

Per il fine-tuning con RL, si può assegnare reward all'output finale (*outcome reward*) o a ogni passo del ragionamento (*process reward*). Lightman et al. (2023) mostrano che il Process Reward Model (PRM) produce un miglioramento di generalizzazione significativo:

$$r_{\text{PRM}}(s_{1:k}, a) = \sum_{j=1}^{k} r_j(s_j, a_j) \qquad \text{(reward per ogni step)}$$

$$r_{\text{ORM}}(s_{1:k}, a) = r(s_k) \qquad \text{(reward solo al termine)}$$

Il PRM guida il modello a produrre passaggi di ragionamento corretti anche quando la risposta finale è sbagliata, riducendo la varianza dell'aggiornamento RL.

PARTE 67

Retrieval-Augmented Generation (RAG)

67.1 Dense Retrieval: DPR

Il **Dense Passage Retrieval** (Karpukhin et al., 2020) usa due encoder BERT separati per query e documenti:

$$\text{sim}(q, d) = E_Q(q)^T E_D(d)$$

L'encoder dei documenti pre-calcola embedding per tutti i documenti del corpus (miliardi di testi) e li indicizza con FAISS (Facebook AI Similarity Search) per la ricerca approssimata dei vicini più prossimi. Il retrieval di top-k documenti è $O(d \log N)$ invece di $O(Nd)$.

Training con in-batch negative

Con un mini-batch di B coppie (q_i, d_i^+) (query + documento positivo), si usano tutti gli altri documenti del batch come negativi:

$$\mathcal{L}_{\text{DPR}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{e^{\text{sim}(q_i, d_i^+)/\tau}}{\sum_{j=1}^{B} e^{\text{sim}(q_i, d_j)/\tau}}$$

Con batch size $B = 128$: 127 negativi per query in modo efficiente (non serve campionare negativi aggiuntivi).

67.2 RAG-Fusion e Fusione dei Documenti Recuperati

Nel RAG standard (Lewis et al., 2020), il modello genera condizionato su query q e top-k documenti $d_1, \ldots, d_k$:

$$P(y|q) = \sum_{i=1}^{k} P_\eta(d_i|q)\, P_\theta(y|q, d_i)$$

La somma marginale su k documenti può essere calcolata in due modi:

- **RAG-Sequence:** per ogni documento d_i si genera una risposta completa, poi si fa beam search globale sulla sequenza
- **RAG-Token:** per ogni token y_t, si marginalizza sui documenti: $P(y_t|y_{

RAG-Token permette al modello di "switchare" documento per diversi token della risposta — utile per domande che sintetizzano informazioni da multiple fonti.

Ricerca vettoriale approssimata: FAISS-IVF

Per corpus con miliardi di vettori, la ricerca esatta è impossibile. FAISS-IVF (Inverted File Index) divide i vettori in n_{list} cluster (k-means), e per ogni query cerca solo nei cluster più vicini:

$$\text{Top-}k \text{ approssimato}(\mathbf{q}) = \text{Top-}k\left(\bigcup_{c \in \text{Top-}n_{\text{probe}}(\mathbf{q})} \text{cluster}(c) \right)$$

Con $n_{\text{probe}} = 64$ su $n_{\text{list}} = 4096$: si esaminano solo il $64/4096 \approx 1.6\%$ dei vettori per query, con recall@100 tipicamente $> 95\%$.

PARTE 68

Modelli per il Parlato: WaveNet, Tacotron e Whisper

68.1 WaveNet: Convoluzione Causale Dilatata

WaveNet (van den Oord et al., 2016) modella la distribuzione del segnale audio $\mathbf{x} = (x_1, \ldots, x_T)$ autoregressivamente:

$$P(\mathbf{x}) = \prod_{t=1}^{T} P(x_t | x_1, \ldots, x_{t-1})$$

Il segnale audio a 16kHz con 16-bit richiede receptive field molto ampi — dell'ordine di migliaia di campioni. WaveNet usa **convoluzione causale dilatata** con fattori di dilatazione $d = 1, 2, 4, 8, \ldots, 512$:

$$(\mathbf{x} \star_d \mathbf{W})[t] = \sum_{k=0}^{K-1} W[k] \cdot x[t - d \cdot k] \qquad \text{(salto di } d \text{ posizioni)}$$

Con $L = 10$ layer e dilatazioni $d = 2^0, 2^1, \ldots, 2^9$, il receptive field è $1 + (K-1)\sum_{l=0}^{9} 2^l = 1 + 1 \cdot (2^{10} - 1) = 1024$ campioni. Impilando M blocchi di 10 layer: receptive field di $M \cdot 1024$ campioni.

Gated Activation Units

WaveNet usa gate di attivazione ispirati alle LSTM:

$$\mathbf{z} = \tanh(\mathbf{W}_{f,k} \star \mathbf{x}) \odot \sigma(\mathbf{W}_{g,k} \star \mathbf{x})$$

dove $\mathbf{W}_{f,k}$ e $\mathbf{W}_{g,k}$ sono i filtri filter e gate del layer k. Questo combina la non-linearità di $\tanh$ con il gating di σ.

Mu-law quantizzazione

WaveNet quantizza l'audio in $\mu + 1 = 256$ livelli con trasformazione mu-law per ridurre il range dinamico:

$$f(x) = \text{sign}(x) \frac{\ln(1 + \mu|x|)}{\ln(1 + \mu)}, \quad \mu = 255$$

68.2 Whisper: Speech Recognition con Transformer

Whisper (Radford et al., 2022) è un modello encoder-decoder per il riconoscimento automatico del parlato (ASR), addestrato su 680.000 ore di audio multilingue.

Preprocessing: Log-Mel Spectrogram

L'audio grezzo viene trasformato in un log-mel spectrogram tramite:

$$X[m, k] = \log \left(\sum_{n=0}^{N-1} x[n + mH] \cdot w[n] \cdot e^{-j2\pi nk/N} \right)^2 \cdot \mathbf{H}_{\text{mel}}[k, :]$$

dove H è il hop size, $w[n]$ la finestra di Hann, e $\mathbf{H}_{\text{mel}}$ la matrice di filterbank mel. Il risultato è un tensore $[80, T/H]$ (80 canali mel, T/H frame temporali). Questo viene proiettato in sequenze di token tramite due layer Conv1D con stride 2.

Architettura e multitask

Whisper usa token speciali per indicare task, lingua e timestamp — un singolo modello gestisce riconoscimento (ASR), traduzione (ST), e identificazione della lingua (LID):

$$\text{input decoder} = \texttt{[SOT]}\ \texttt{[lang]}\ \texttt{[task]}\ \texttt{[notimestamp]}\ \text{trascrizione}\ \texttt{[EOT]}$$

La loss è la cross-entropy standard su tutti i token della trascrizione. Il multitask training agisce come regolarizzatore e migliora la robustezza cross-lingue.

PARTE 69

Neural Architecture Search (NAS)

69.1 DARTS: Differentiable Architecture Search

DARTS (Liu et al., 2019) rende la ricerca dell'architettura differenziabile relaxando lo spazio discreto delle operazioni in uno spazio continuo. Per ogni arco (i, j) in una cella, invece di scegliere un'operazione, si usa una media pesata di tutte le operazioni candidate $\mathcal{O}$:

$$\bar{o}^{(i,j)}(\mathbf{x}) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} \cdot o(\mathbf{x})$$

I parametri architetturali $\boldsymbol{\alpha} = \{\alpha_o^{(i,j)}\}$ determinano l'importanza di ogni operazione. L'ottimizzazione alterna tra:

- Aggiornamento dei **pesi della rete** $\mathbf{w}$ su training set: $\nabla_{\mathbf{w}} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\alpha})$
- Aggiornamento dei **parametri architetturali** $\boldsymbol{\alpha}$ su validation set: $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha})$

Il gradiente di $\boldsymbol{\alpha}$ attraverso $\mathbf{w}^*(\boldsymbol{\alpha})$ richiede la differenziazione di secondo ordine. DARTS usa l'approssimazione:

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}^*, \boldsymbol{\alpha}) \approx \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w} - \xi \nabla_{\mathbf{w}} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\alpha}), \boldsymbol{\alpha})$$

Espandendo al primo ordine in ξ: $\approx \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}, \boldsymbol{\alpha}) - \xi \nabla^2_{\boldsymbol{\alpha}, \mathbf{w}} \mathcal{L}_{\text{train}} \cdot \nabla_{\mathbf{w}} \mathcal{L}_{\text{val}}$.

PARTE 70

Hypernetworks

Reti neurali che generano i pesi di altre reti neurali: una meta-architettura con applicazioni al continual learning, personalizzazione e compressione.

70.1 Definizione e Proprietà

Un **hypernetwork** (Ha et al., 2017) è una rete h_ϕ che genera i pesi θ di una rete target f_θ:

$$\theta = h_\phi(\mathbf{z}), \qquad \hat{y} = f_{h_\phi(\mathbf{z})}(\mathbf{x})$$

dove $\mathbf{z}$ è un vettore di input dell'hypernetwork (es. embedding del task, del soggetto, del dominio). L'hypernetwork ha $|\phi|$ parametri e genera $|\theta|$ parametri per ogni $\mathbf{z}$.

Vantaggi teorici

Un hypernetwork che mappa $\mathbf{z} \in \mathbb{R}^k$ in $\theta \in \mathbb{R}^{|\theta|}$ con $k \ll |\theta|$ comprime l'informazione: le variazioni del comportamento della rete target sono parametrizzate in uno spazio di dimensione k molto più piccola. Questo ha due conseguenze:

- **Generalizzazione:** nuovi task richiedono solo la ricerca di $\mathbf{z}^*$ ottimale, con ottimizzazione in $\mathbb{R}^k$ invece di $\mathbb{R}^{|\theta|}$
- **Continual learning:** nuovi task cambiano solo $\mathbf{z}$, non ϕ — i task precedenti restano accessibili con i loro $\mathbf{z}$ precedenti

Connessione con LoRA

LoRA (Sezione 35 del Vol. I) può essere interpretato come un hypernetwork degenere: $\Delta\mathbf{W} = h_\phi(\mathbf{z}) = \mathbf{BA}$ con parametri $\mathbf{B}, \mathbf{A}$ fissi (non condizionati). Un hypernetwork condizionato sul task generalizza LoRA a aggiornamenti dipendenti dal contesto.

PARTE 71

Capsule Networks

Rappresentare pose e relazioni geometriche tramite capsule — vettori di attività che codificano sia la presenza che le proprietà di un'entità.

71.1 Capsule e Agreement Routing

Una **capsula** è un gruppo di neuroni il cui *vettore* di attività rappresenta i parametri di istanziazione di un'entità (posizione, orientamento, scala, deformazione). La lunghezza del vettore rappresenta la probabilità di presenza dell'entità.

Squashing function

La funzione non-lineare di squashing mantiene la direzione del vettore ma ne scala la lunghezza a $(0, 1)$:

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$$

dove $\mathbf{s}_j = \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$ è la somma pesata delle "prediction vectors" $\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i$.

Dynamic Routing by Agreement

I coefficienti di routing c_{ij} sono determinati iterativamente: se la predizione $\hat{\mathbf{u}}_{j|i}$ di capsula bassa i è in accordo (grande prodotto scalare) con l'output $\mathbf{v}_j$ di capsula alta j, allora c_{ij} aumenta:

$$b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$$

$$c_{ij} = \frac{e^{b_{ij}}}{\sum_k e^{b_{ik}}}$$

L'algoritmo alterna l'aggiornamento di $\mathbf{v}_j$ (dato c_{ij}) e l'aggiornamento di c_{ij} (dato $\mathbf{v}_j$) per r iterazioni. È analogo all'E-step e M-step dell'algoritmo EM, ma differenziabile rispetto ai pesi $\mathbf{W}_{ij}$.

PARTE 72

Spiking Neural Networks (SNN)

Reti neurali biologicamente plausibili che comunicano tramite spike (impulsi binari), con applicazioni all'edge computing ultra-efficiente.

72.1 Modello Leaky Integrate-and-Fire (LIF)

Il neurone LIF integra i correnti di input nella sua membrana potenziale $u(t)$ e genera uno spike quando supera la soglia ϑ:

$$\tau_m \frac{du}{dt} = -u(t) + R\,I(t)$$
$$\text{se } u(t) \geq \vartheta: \quad u(t) \leftarrow u_r, \text{ emit spike}$$

dove $\tau_m = RC$ è la costante di tempo della membrana, R la resistenza, C la capacità e u_r il potenziale di reset. La dinamica discretizzata con passo Δt:

$$u[t+1] = \beta u[t] + (1-\beta) I[t], \quad \beta = e^{-\Delta t/\tau_m} \approx 1 - \Delta t/\tau_m$$
$$s[t+1] = \mathbf{1}[u[t+1] \geq \vartheta], \quad u[t+1] \leftarrow u[t+1](1 - s[t+1]) + u_r s[t+1]$$

Il problema del gradiente con gli spike

La funzione di Heaviside $s = \mathbf{1}[u \geq \vartheta]$ ha derivata zero quasi ovunque — non differenziabile. Si usa il **surrogate gradient**: durante il backward pass si sostituisce la derivata con una funzione continua:

$$\frac{\partial s}{\partial u} \approx \frac{1}{\pi} \frac{1}{1 + (u - \vartheta)^2 \pi^2} \quad \text{(surrogate Arctangente)}$$
$$\text{oppure: } \frac{\partial s}{\partial u} \approx \max(0, 1 - |u - \vartheta|) \quad \text{(surrogate triangolare)}$$

72.2 STDP: Spike-Timing Dependent Plasticity

La regola di apprendimento biologica STDP aggiorna la sinapsi in base alla differenza temporale $\Delta t = t_{\text{post}} - t_{\text{pre}}$:

$$\Delta w = \begin{cases} A_+ \exp(-\Delta t / \tau_+) & \Delta t > 0 \text{ (pre precede post: LTP)} \\ -A_- \exp(\Delta t / \tau_-) & \Delta t < 0 \text{ (post precede pre: LTD)} \end{cases}$$

LTP (Long-Term Potentiation) rafforza la sinapsi se il neurone pre sinaptico contribuisce all'attivazione del post; LTD (Long-Term Depression) la indebolisce altrimenti. Questa regola è una forma di Hebbian learning biologicamente plausibile: "neuroni che si attivano insieme si connettono insieme".

PARTE 73

Quantum Machine Learning

Fondamenti della computazione quantistica applicata all'apprendimento automatico: qubit, circuiti variazionali e il potenziale (e i limiti) del vantaggio quantistico.

73.1 Qubit e Spazio di Hilbert

Un qubit è un sistema quantistico a due livelli il cui stato è un vettore in $\mathbb{C}^2$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1, \quad \alpha, \beta \in \mathbb{C}$$

dove $|0\rangle = (1,0)^T$ e $|1\rangle = (0,1)^T$ sono i vettori della base computazionale. La notazione bra-ket è: $|\psi\rangle$ (ket, vettore colonna), $\langle\psi|$ (bra, vettore riga coniugato), $\langle\phi|\psi\rangle$ (prodotto scalare complesso). Per n qubit: lo stato vive in $\mathbb{C}^{2^n}$ — spazio di Hilbert di dimensione esponenziale.

Gate quantistici

Le operazioni su qubit sono matrici unitarie U ($U^\dagger U = I$):

Gate	Matrice	Effetto
Hadamard H	$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$	$\|0\rangle \to \frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$ (sovrapposizione uniforme)
Pauli-X	$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$	NOT quantistico: $\|0\rangle \leftrightarrow \|1\rangle$
Rotazione $R_z(\theta)$	$\begin{pmatrix}e^{-i\theta/2} & 0\\ 0 & e^{i\theta/2}\end{pmatrix}$	Rotazione di θ attorno all'asse z della sfera di Bloch
CNOT	$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}$	Flip del qubit target se il qubit controllo è $\|1\rangle$ (entanglement)

73.2 Variational Quantum Circuits (VQC)

Un VQC è un circuito quantistico $U(\boldsymbol{\theta})$ con parametri $\boldsymbol{\theta} \in \mathbb{R}^p$ appresi classicamente. La struttura di base è:

$$U(\boldsymbol{\theta}) = \prod_{l=1}^{L} U_l(\theta_l)\, W_l$$

dove W_l sono gate fissi (es. CNOT per l'entanglement) e $U_l(\theta_l) = e^{-i\theta_l H_l/2}$ sono rotazioni parametrizzate (H_l è un operatore hermitiano, tipicamente Pauli). La funzione obiettivo è il valore atteso di un osservabile $\hat{O}$:

$$f(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | \hat{O} | \psi(\boldsymbol{\theta}) \rangle, \quad |\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|0\rangle^{\otimes n}$$

Parameter-Shift Rule: gradiente esatto

Per rotazioni della forma $U(\theta) = e^{-i\theta H/2}$, il gradiente rispetto a θ è calcolabile con misurazioni:

PARAMETER-SHIFT RULE

$$\frac{\partial f}{\partial \theta_k} = \frac{1}{2}\left[f(\boldsymbol{\theta} + \frac{\pi}{2}\mathbf{e}_k) - f(\boldsymbol{\theta} - \frac{\pi}{2}\mathbf{e}_k) \right]$$

Questo è il gradiente *esatto* (non approssimato), calcolabile con solo 2 valutazioni del circuito per ogni parametro. Analogo alle finite difference ma con errore zero (per i gate che soddisfano la struttura parametrica).

Barren Plateaus

Con n qubit e profondità L, il gradiente di un VQC con inizializzazione casuale decade esponenzialmente:

$$\mathrm{Var}\left[\frac{\partial f}{\partial \theta_k}\right] = O\left(\frac{1}{2^n}\right) \quad \text{per circuiti 2-design}$$

Questo è il **barren plateau problem**: con molti qubit, il paesaggio di ottimizzazione diventa piatto — il gradiente è esponenzialmente piccolo nella maggior parte delle direzioni. È l'analogo

quantistico del vanishing gradient, e costituisce la sfida principale per l'addestramento di VQC profondi.

PARTE 74

Optimal Control e Principio di Pontryagin

La teoria del controllo ottimale come fondamento matematico per le Neural ODE, il Reinforcement Learning e l'addestramento di reti ricorrenti.

74.1 Il Principio del Massimo di Pontryagin

Un problema di controllo ottimale in tempo continuo è:

$$\min_{\mathbf{u}(\cdot)} \int_0^T \ell(\mathbf{x}(t), \mathbf{u}(t))\, dt + \phi(\mathbf{x}(T))$$
$$\text{s.t. } \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0$$

dove $\mathbf{x}(t) \in \mathbb{R}^n$ è lo stato, $\mathbf{u}(t) \in \mathcal{U}$ il controllo, ℓ il costo istantaneo e ϕ il costo terminale. L'**Hamiltoniana del controllo** è:

$$\mathcal{H}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = \ell(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^T f(\mathbf{x}, \mathbf{u})$$

dove $\boldsymbol{\lambda}(t) \in \mathbb{R}^n$ è la variabile aggiunta (costate).

PRINCIPIO DEL MASSIMO DI PONTRYAGIN (PMP)

Se $(\mathbf{x}^*, \mathbf{u}^*)$ è un controllo ottimale, esiste $\boldsymbol{\lambda}^*(t)$ tale che:

$$\dot{\mathbf{x}}^* = \frac{\partial \mathcal{H}}{\partial \boldsymbol{\lambda}} = f(\mathbf{x}^*, \mathbf{u}^*) \qquad \text{(equazione di stato)}$$

$$\dot{\boldsymbol{\lambda}}^* = -\frac{\partial \mathcal{H}}{\partial \mathbf{x}} = -\frac{\partial \ell}{\partial \mathbf{x}} - \left(\frac{\partial f}{\partial \mathbf{x}}\right)^T \boldsymbol{\lambda}^* \qquad \text{(equazione costate)}$$

$$\mathbf{u}^*(t) = \arg\min_{\mathbf{u} \in \mathcal{U}} \mathcal{H}(\mathbf{x}^*(t), \mathbf{u}, \boldsymbol{\lambda}^*(t)) \qquad \text{(condizione di minimizzazione)}$$

con condizione terminale $\boldsymbol{\lambda}^*(T) = \frac{\partial \phi}{\partial \mathbf{x}}(\mathbf{x}^*(T))$.

Connessione con Neural ODE e Adjoint Method

Il metodo adjoint per le Neural ODE (Sezione 34 del Vol. I) è esattamente il PMP applicato al training: lo stato $\mathbf{x}$ è l'attivazione, il controllo $\mathbf{u} = \theta$ sono i parametri della rete, il costo è $\mathcal{L}$, e la variabile aggiunta $\boldsymbol{\lambda} = \partial \mathcal{L} / \partial \mathbf{x}$ è il gradiente. L'equazione costate è la backpropagation continua.

74.2 Equazione di Hamilton-Jacobi-Bellman

La funzione valore ottimale $V(\mathbf{x}, t) = \min_{\mathbf{u}} \int_t^T \ell \, ds + \phi(\mathbf{x}(T))$ soddisfa la PDE di Hamilton-Jacobi-Bellman:

$$-\frac{\partial V}{\partial t} = \min_{\mathbf{u} \in \mathcal{U}} \left[\ell(\mathbf{x}, \mathbf{u}) + \nabla_{\mathbf{x}} V \cdot f(\mathbf{x}, \mathbf{u}) \right]$$

con condizione terminale $V(\mathbf{x}, T) = \phi(\mathbf{x})$. La policy ottimale è:

$$\mathbf{u}^*(\mathbf{x}, t) = \arg\min_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \nabla_{\mathbf{x}} V(\mathbf{x}, t) \cdot f(\mathbf{x}, \mathbf{u}) \right]$$

L'equazione di Bellman del Reinforcement Learning (Sezione 16 del Vol. I) è la versione discreta dell'HJB.

PARTE 75

Teoria del Campionamento e Aliasing nelle Reti Neurali

Il teorema di Nyquist-Shannon come fondamento per la scelta delle architetture CNN, la connessione tra pooling e aliasing, e le conseguenze per l'equivarianza.

75.1 Teorema di Nyquist-Shannon

TEOREMA DI NYQUIST-SHANNON (CAMPIONAMENTO)

Un segnale $x(t)$ a banda limitata con frequenza massima $f_{\max}$ può essere ricostruito esattamente dai suoi campioni $x[n] = x(nT_s)$ se e solo se la frequenza di campionamento soddisfa:

$$f_s = \frac{1}{T_s} \geq 2f_{\max} \qquad \text{(frequenza di Nyquist)}$$

La ricostruzione avviene tramite interpolazione sinc:

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \cdot \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right), \quad \operatorname{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$$

75.2 Aliasing nel Pooling e Convoluzione con Stride

Il max pooling con stride $s > 1$ effettua un campionamento del segnale (feature map) senza pre-filtrare — se la feature map contiene componenti ad alta frequenza, si produce **aliasing**: frequenze alte vengono confuse con frequenze basse, distorcendo la risposta.

Zhang (2019) mostra che questo causa una mancanza di equivarianza per traslazione nelle CNN: traslare l'input di un pixel può produrre output molto diversi. La soluzione: applicare un filtro passa-basso (blurring) prima del campionamento:

$$\text{BlurPool}(\mathbf{x}) = (\mathbf{x} \star \mathbf{h})[::s]$$

dove $\mathbf{h}$ è un filtro gaussiano o binomiale e $[::s]$ denota il campionamento con stride s. Il filtraggio rimuove le frequenze superiori a $f_s/2$ prima del downsampling, prevenendo l'aliasing.

Connessione con la Teoria di Fourier Discreta

La DFT di una sequenza di lunghezza N: $\hat{X}[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$ per $k = 0, \ldots, N-1$. Per una feature map di dimensione N campionata con stride s, la DFT dell'output ha le frequenze k/N con $k \in \{0, \ldots, N/s - 1\}$. Le componenti con $k > N/(2s)$ creano aliasing:

$$\hat{X}_{\text{aliased}}[k] = \hat{X}[k] + \hat{X}[k + N/s] + \hat{X}[k + 2N/s] + \cdots$$

Il filtro anti-aliasing sopprime queste componenti prima del campionamento, garantendo $\hat{X}[k + mN/s] \approx 0$ per $m \geq 1$.

75.3 Equivarianza e Aliasing

Una rete CNN è **equivariante per traslazione** se $f(T_\delta \mathbf{x}) = T_\delta f(\mathbf{x})$ dove T_δ è la traslazione di δ pixel. Le convoluzioni sono equivarianti per costruzione; ma il pooling con stride viola questa proprietà a causa dell'aliasing. La BlurPool recupera quasi completamente l'equivarianza misurata come:

$$\text{Eq}(f) = 1 - \mathbb{E}_{\mathbf{x}, \delta}\left[\frac{\| f(T_\delta \mathbf{x}) - T_\delta f(\mathbf{x}) \|}{\| f(\mathbf{x}) \|}\right]$$

Zhang (2019) riporta miglioramenti dall'86% all'99% di equivarianza su ResNet con BlurPool.

PARTE 76

Score-Based Generative Models: Framework SDE Unificato

Song et al. (2021) unificano i modelli di diffusione e lo score matching in un unico framework stocastico, consentendo campionamento flessibile tramite ODE deterministiche e connessioni profonde con la meccanica statistica.

76.1 SDE Forward e Reverse-Time

Il processo di diffusione in avanti è una SDE (Stochastic Differential Equation) generale:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\, dt + g(t)\, d\mathbf{w}$$

dove $\mathbf{f} : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ è il drift, $g : [0, T] \to \mathbb{R}$ è il coefficiente di diffusione scalare, e $\mathbf{w}$ è il moto Browniano standard in $\mathbb{R}^d$. L'equazione descrive come i dati $\mathbf{x}(0) \sim p_{\text{data}}$ si trasformano progressivamente in rumore $\mathbf{x}(T) \approx p_T$ (solitamente gaussiana standard).

Il risultato fondamentale (Anderson, 1982) è che esiste una SDE in tempo inverso con le stesse distribuzioni marginali:

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt + g(t)\, d\bar{\mathbf{w}}$$

dove $d\bar{\mathbf{w}}$ è il moto Browniano in tempo inverso (da T a 0) e $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ è lo **score function** — il gradiente del log-density della distribuzione marginale al tempo t.

Casi speciali: VP-SDE e VE-SDE

La **VP-SDE** (Variance Preserving) corrisponde a DDPM:

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\, dt + \sqrt{\beta(t)}\, d\mathbf{w}$$

con $\beta(t)$ che varia linearmente da $\beta_{\min}$ a $\beta_{\max}$. La distribuzione condizionata è $p_{0t}(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(t); e^{-\frac{1}{2}\int_0^t \beta(s)ds}\mathbf{x}(0), (1 - e^{-\int_0^t \beta(s)ds})\mathbf{I})$.

La **VE-SDE** (Variance Exploding) corrisponde a SMLD (Score Matching with Langevin Dynamics):

$$d\mathbf{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}}\, d\mathbf{w}, \qquad p_{0t}(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(t); \mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)]\mathbf{I})$$

76.2 Obiettivo di Training Continuo

Si addestra una rete $\mathbf{s}_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. Poiché p_t è intractabile, si usa la score matching denoising: per la VP-SDE, lo score analitico della distribuzione condizionata è:

$$\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t)|\mathbf{x}(0)) = -\frac{\mathbf{x}(t) - e^{-\frac{1}{2}\bar{\beta}(t)}\mathbf{x}(0)}{1 - e^{-\bar{\beta}(t)}} = -\frac{\boldsymbol{\varepsilon}}{\sqrt{1 - e^{-\bar{\beta}(t)}}}$$

dove $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ è il rumore aggiunto e $\bar{\beta}(t) = \int_0^t \beta(s) ds$. La loss pesata per tempi $t \sim \mathcal{U}[0, T]$:

$$\mathcal{L}_{\text{SDE}} = \mathbb{E}_{t, \mathbf{x}(0), \mathbf{x}(t)}\left[\lambda(t) \|\mathbf{s}_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t)|\mathbf{x}(0))\|^2\right]$$

Con la scelta $\lambda(t) = 1 - e^{-\bar{\beta}(t)}$, la loss si semplifica a quella di DDPM (predizione del rumore).

76.3 Probability Flow ODE e Stima della Likelihood

Ogni SDE ha una corrispondente **ODE deterministica** — la Probability Flow ODE — che mantiene le stesse marginali p_t:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

Vantaggi rispetto alla SDE: (1) **determinismo**: stessa immagine da stesso rumore iniziale; (2) **invertibilità esatta**: dato $\mathbf{x}(0)$ si recupera $\mathbf{x}(T)$ con precisione arbitraria; (3) **stima della likelihood** tramite il change-of-variables formula per ODE (Instantaneous Change of Variables):

$$\log p_0(\mathbf{x}(0)) = \log p_T(\mathbf{x}(T)) + \int_0^T \operatorname{tr}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} - \frac{1}{2} g^2 \frac{\partial \mathbf{s}_\theta}{\partial \mathbf{x}}\right) dt$$

La traccia del Jacobiano si stima con lo stimatore di Hutchinson: $\operatorname{tr}(\mathbf{A}) \approx \boldsymbol{\varepsilon}^T \mathbf{A} \boldsymbol{\varepsilon}$ con $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (valore atteso esatto, varianza controllata).

76.4 Classifier-Free Guidance: Derivazione Completa

La guida classica (classifier guidance) modifica lo score con il gradiente di un classificatore addestrato su dati rumorosi: $\tilde{\mathbf{s}} = \mathbf{s}_\theta(\mathbf{x}, t) + w \nabla_{\mathbf{x}} \log p_\phi(y | \mathbf{x}, t)$. Il problema: richiede un classificatore separato robusto al rumore.

Il **Classifier-Free Guidance** (CFG, Ho e Salimans, 2022) elimina questo requisito. Si addestra un singolo modello $\mathbf{s}_\theta(\mathbf{x}, t, y)$ che riceve la label y come condizionamento, con azzeramento casuale ($y \leftarrow \emptyset$ con prob. $p_{\text{uncond}} \approx 0.1$). La guida a inferenza:

$$\tilde{\mathbf{s}}_\theta(\mathbf{x}, t, y) = \underbrace{\mathbf{s}_\theta(\mathbf{x}, t, \emptyset)}_{\text{non condizionato}} + w \cdot \underbrace{\left[\mathbf{s}_\theta(\mathbf{x}, t, y) - \mathbf{s}_\theta(\mathbf{x}, t, \emptyset)\right]}_{\text{guida}}$$

Riscrivendo: $\tilde{\mathbf{s}} = (1 + w)\mathbf{s}_\theta(\mathbf{x}, t, y) - w\mathbf{s}_\theta(\mathbf{x}, t, \emptyset)$. Per $w = 0$: campionamento standard. Per $w > 0$: il modello si allontana dalla distribuzione non condizionata verso quella condizionata, con guadagno di fedeltà al costo di diversità.

Interpretazione in termini di KL

La guida con peso w campiona dalla distribuzione $\tilde{p}(\mathbf{x}|y) \propto p(\mathbf{x}|y)^{1+w} \, p(\mathbf{x})^{-w}$. Per $w > 0$, questa distribuzione è più "concentrata" di $p(\mathbf{x}|y)$ — riduce la varianza a scapito della diversità. Il trade-off precisione-recall:

$$\text{Precisione} \propto \mathbb{E}_y[D_{\text{KL}}(\tilde{p}(\cdot|y) \| p(\cdot|y))] \downarrow \text{ con } w \uparrow$$

$$\text{Recall} \propto D_{\text{KL}}(p(\cdot|y) \| \tilde{p}(\cdot|y)) \uparrow \text{ con } w \uparrow$$

PARTE 77

Reti Neurali Equivarianti e Teoria dei Gruppi

Come costruire reti che rispettano simmetrie fisiche o geometriche tramite la teoria matematica dei gruppi di Lie e delle rappresentazioni.

77.1 Gruppi di Lie e Rappresentazioni

Un **gruppo di Lie** G è un gruppo con struttura di varietà differenziabile — l'operazione di gruppo è differenziabile. Esempi fondamentali:

Gruppo	Dimensione	Elementi tipici	Applicazione ML
$SO(2)$	1	Rotazioni 2D: $\begin{pmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{pmatrix}$	Feature equivarianti per immagini rotanti
$SO(3)$	3	Rotazioni 3D (matrici 3×3 ortogonali, det=1)	Molecole, proteine, nuvole di punti
$SE(3)$	6	Rotazioni + traslazioni 3D	Simulazioni fisiche, robotica
$U(1)$	1	Fase complessa $e^{i\theta}$	Quantum ML
S_n	$n!$ (discreto)	Permutazioni di n oggetti	Insiemi, grafi

Una **rappresentazione** di G in $\mathbb{R}^d$ è un omomorfismo $\rho : G \to GL(\mathbb{R}^d)$ (mappe lineari invertibili). Ogni $g \in G$ corrisponde a una matrice $\rho(g) \in \mathbb{R}^{d\times d}$ con $\rho(g_1 g_2) = \rho(g_1)\rho(g_2)$.

77.2 Equivarianza Formale

Una funzione $f : \mathcal{X} \to \mathcal{Y}$ è **equivariante** rispetto al gruppo G con rappresentazioni $\rho_{\mathcal{X}}$ e $\rho_{\mathcal{Y}}$ se:

$$f(\rho_{\mathcal{X}}(g)\cdot \mathbf{x}) = \rho_{\mathcal{Y}}(g)\cdot f(\mathbf{x}) \qquad \forall g \in G,\, \mathbf{x}\in\mathcal{X}$$

La **invarianza** è il caso $\rho_{\mathcal{Y}}(g) = \mathbf{I}$ per tutti i g — l'output non cambia sotto trasformazioni del gruppo. Esempi:

- La convoluzione standard è equivariante per traslazioni: $[f \star g](T_\delta \mathbf{x}) = T_\delta [f \star g](\mathbf{x})$
- La max pooling globale è invariante per permutazioni degli elementi del feature map
- La norma $\|\mathbf{x}\|$ è invariante per rotazioni: $\|R\mathbf{x}\| = \|\mathbf{x}\|$ per $R \in SO(3)$

77.3 G-CNN: Group-Equivariant Convolutional Networks

Cohen e Welling (2016) definiscono la **convoluzione su gruppo** per funzioni $f : G \to \mathbb{R}^C$ e filtri $\psi : G \to \mathbb{R}^C$:

$$[f \star_G \psi](g) = \sum_{h \in G} \sum_{c=1}^{C} f_c(h)\, \psi_c(g^{-1} h)$$

Questa operazione è equivariante per traslazioni nel gruppo: $[(L_{g_0} f) \star_G \psi](g) = [f \star_G \psi](g_0^{-1} g)$ dove $L_{g_0} f(g) = f(g_0^{-1} g)$.

Esempio: Gruppo $p4$ (traslazioni + rotazioni di 90°)

Il gruppo $p4$ ha elementi (r, t) con $r \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$ e $t \in \mathbb{Z}^2$. Una G-CNN su $p4$ produce feature map in $\mathbb{R}^{4 \times H \times W \times C}$ — quattro copie dell'immagine rotata di $0^\circ, 90^\circ, 180^\circ, 270^\circ$. Con C filtri in ogni layer, la rete ha la stessa espressività di una CNN con $4C$ filtri, ma garantisce equivarianza per rotazioni di multipli di 90°.

77.4 EGNN: Equivariant Graph Neural Networks

EGNN (Satorras et al., 2021) costruisce GNN equivarianti rispetto a $E(3)$ (rotazioni, riflessioni, traslazioni) per sistemi molecolari e fisici. Ogni nodo i ha feature scalari $\mathbf{h}_i \in \mathbb{R}^d$ e coordinate 3D $\mathbf{x}_i \in \mathbb{R}^3$. Le equazioni di aggiornamento:

1. $\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i, \mathbf{h}_j, \|\mathbf{x}_i - \mathbf{x}_j\|^2, a_{ij})$ dove a_{ij} sono attributi dell'arco
   Il messaggio usa solo distanze (scalari invarianti), non coordinate assolute
2. $\mathbf{x}_i' = \mathbf{x}_i + \sum_{j \neq i} (\mathbf{x}_i - \mathbf{x}_j)\, \phi_x(\mathbf{m}_{ij})$
   Aggiornamento delle coordinate: solo differenze $(\mathbf{x}_i - \mathbf{x}_j)$ — equivariante per $E(3)$
3. $\mathbf{h}_i' = \phi_h(\mathbf{h}_i, \sum_j w_{ij}\mathbf{m}_{ij})$ con $w_{ij} = \phi_w(\mathbf{m}_{ij})$
   Feature scalari: invarianti per $E(3)$ (non cambiano con rotazioni/traslazioni)

Prova dell'equivarianza

Per una rotazione $R \in SO(3)$, se si applicano $\mathbf{x}_i \leftarrow R\mathbf{x}_i$ per tutti i nodi, allora $\|R\mathbf{x}_i - R\mathbf{x}_j\| = \|\mathbf{x}_i - \mathbf{x}_j\|$ — le distanze non cambiano, quindi $\mathbf{m}_{ij}$ non cambia, quindi $\mathbf{h}_i'$ non cambia. Per le coordinate: $R\mathbf{x}_i + \sum_j (R\mathbf{x}_i - R\mathbf{x}_j)\phi_x(\mathbf{m}_{ij}) = R(\mathbf{x}_i + \sum_j (\mathbf{x}_i - \mathbf{x}_j)\phi_x(\mathbf{m}_{ij})) = R\mathbf{x}_i'$ — le coordinate si trasformano coerentemente con la rotazione. □

PARTE 78

Flow Matching e Stochastic Interpolants

Un'alternativa ai modelli di diffusione basata su interpolazione di traiettorie: training più semplice, inferenza più rapida, e generalizzazione alle traiettorie non-lineari.

78.1 Conditional Flow Matching

Lipman et al. (2022) propongono di apprendere un campo vettoriale $v_\theta(\mathbf{x}, t)$ che genera il flusso ottimale da p_0 (gaussiana) a p_1 (dati):

$$\frac{d\mathbf{x}}{dt} = v_\theta(\mathbf{x}, t), \qquad \mathbf{x}(0) \sim p_0(\mathbf{x}), \quad \mathbf{x}(1) \sim p_1(\mathbf{x})$$

Il campo vettoriale target "marginale" u_t è difficile da calcolare direttamente. Il **Conditional Flow Matching** condiziona sulle coppie $(\mathbf{x}_0, \mathbf{x}_1)$ e usa traiettorie lineari condizionate:

$$\psi_t(\mathbf{x}|\mathbf{x}_1) = (1 - (1 - \sigma_{\min})t)\mathbf{x} + t\mathbf{x}_1, \qquad u_t(\mathbf{x}|\mathbf{x}_1) = \mathbf{x}_1 - (1 - \sigma_{\min})\mathbf{x}$$

dove $\mathbf{x} = \mathbf{x}_0 \sim p_0$ e $\sigma_{\min}$ è il rumore residuo all'arrivo. La loss:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t \sim \mathcal{U}[0,1], \mathbf{x}_1 \sim p_1, \mathbf{x}_0 \sim p_0}\left[\| v_\theta(\psi_t(\mathbf{x}_0|\mathbf{x}_1), t) - u_t(\mathbf{x}_0|\mathbf{x}_1) \|^2\right]$$

Con traiettorie lineari, il campo vettoriale target è costante lungo ogni traiettoria — la rete deve semplicemente predire la direzione $(\mathbf{x}_1 - \mathbf{x}_0)$ corretta per ogni punto (x_t, t).

Connessione con la diffusione

DDPM corrisponde a CFM con traiettorie non-lineari. Con $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}$, il campo vettoriale target è:

$$u_t(\mathbf{x}_t|\mathbf{x}_0) = \frac{d\mathbf{x}_t}{dt}\Big|_{\mathbf{x}_0} = \frac{\dot{\bar{\alpha}}_t}{2\sqrt{\bar{\alpha}_t}}\mathbf{x}_0 + \frac{\dot{\bar{\alpha}}_t}{2\sqrt{1-\bar{\alpha}_t}}(-\boldsymbol{\varepsilon}) = \frac{\dot{\bar{\alpha}}_t}{2\bar{\alpha}_t}\mathbf{x}_t - \frac{\dot{\bar{\alpha}}_t}{2\sqrt{(1-\bar{\alpha}_t)\bar{\alpha}_t}}\boldsymbol{\varepsilon}$$

La predizione del rumore $\boldsymbol{\varepsilon}_\theta$ di DDPM è equivalente alla predizione del campo vettoriale v_θ con una semplice riscalatura. Il vantaggio del CFM con traiettorie lineari: le traiettorie non si incrociano quasi mai — la rete ha un compito più semplice da apprendere.

78.2 OT-CFM: Flow Matching con Trasporto Ottimale

Il CFM standard accoppia $(\mathbf{x}_0, \mathbf{x}_1)$ in modo indipendente. Con l'**Optimal Transport CFM**, si trovano le coppie che minimizzano il costo quadratico $W_2^2(p_0, p_1)$:

> $$(\mathbf{x}_0^*, \mathbf{x}_1^*) = \arg\min_{\pi} \mathbb{E}_{(\mathbf{x}_0,\mathbf{x}_1)\sim\pi} \|\mathbf{x}_0 - \mathbf{x}_1\|^2$$

L'accoppiamento OT produce traiettorie più dritte (minore curvatura totale), permettendo di campionare con meno passi ODE a parità di qualità. In pratica, si usa il mini-batch OT (Sinkhorn al livello del batch) come approssimazione.

PARTE 79

Teoria della Misura: Fondamenti Rigorosi dell'Apprendimento Statistico

79.1 Spazi di Misura e Integrale di Lebesgue

Uno **spazio di misura** è la tripla $(\Omega, \mathcal{F}, \mu)$ dove: Ω è lo spazio campione (insieme degli esiti possibili); $\mathcal{F}$ è una **σ-algebra** su Ω (famiglia di insiemi "misurabili", chiusa per complementi e unioni numerabili); $\mu : \mathcal{F} \to [0, +\infty]$ è una **misura** (σ-additiva: $\mu(\bigcup_n A_n) = \sum_n \mu(A_n)$ per A_n disgiunti).

Per uno spazio di probabilità: $\mu(\Omega) = 1$ e si scrive P invece di μ. L'**integrale di Lebesgue** di una funzione misurabile $f : \Omega \to \mathbb{R}$:

$$\int_\Omega f \, dP = \sup\left\{ \int_\Omega s \, dP : s \text{ semplice}, \, 0 \le s \le f \right\}$$

dove una funzione semplice è $s = \sum_{k=1}^{n} a_k \mathbf{1}_{A_k}$ con $a_k \ge 0$ e $A_k \in \mathcal{F}$ disgiunti. Il valore atteso di una variabile aleatoria $X : \Omega \to \mathbb{R}$ è $\mathbb{E}[X] = \int_\Omega X \, dP$.

Teorema di convergenza dominata (Lebesgue)

TEOREMA DI CONVERGENZA DOMINATA

Se $\{f_n\}$ è una sequenza di funzioni misurabili con $f_n \to f$ puntualmente q.c., e esiste una funzione integrabile g tale che $|f_n| \le g$ q.c. per ogni n, allora:

$$\lim_{n \to \infty} \int f_n \, dP = \int f \, dP$$

Questo teorema giustifica lo scambio di limite e integrale nelle derivazioni di ML (es. nell'analisi della convergenza degli ottimizzatori e della legge dei grandi numeri).

79.2 Tipi di Convergenza delle Variabili Aleatorie

Per una sequenza $X_1, X_2, \ldots$ di variabili aleatorie e una variabile limite X:

$$\text{q.c.} : P\Big(\lim_{n\to\infty} X_n = X\Big) = 1$$

$$L^p : \lim_{n\to\infty} \mathbb{E}[|X_n - X|^p] = 0 \quad (p \geq 1)$$

$$\text{in prob.} : \forall \varepsilon > 0,\ \lim_{n\to\infty} P(|X_n - X| > \varepsilon) = 0$$

$$\text{in dist.} : \lim_{n\to\infty} F_n(x) = F(x)\ \forall x \text{ con } F \text{ continua}$$

Le implicazioni: q.c. $\Rightarrow$ in prob. $\Rightarrow$ in dist.; $L^p \Rightarrow L^q$ per $p \geq q$; $L^p \Rightarrow$ in prob. Nessuna delle controimplicazioni vale in generale.

Legge dei Grandi Numeri — versioni

Per $X_1, X_2, \ldots$ i.i.d. con $\mathbb{E}[X_i] = \mu$:

$$\text{LGN debole: } \bar{X}_n \xrightarrow{P} \mu \quad (\text{convergenza in probabilità})$$

$$\text{LGN forte: } \bar{X}_n \xrightarrow{\text{q.c.}} \mu \quad (\text{convergenza quasi certa})$$

La LGN forte richiede $\mathbb{E}[|X_1|] < \infty$; la debole solo $\mathbb{E}[X_1^2] < \infty$ (varianza finita). Per il ML: la LGN forte giustifica che il rischio empirico converge al rischio vero per $N \to \infty$ (consistenza del metodo dei momenti e di MLE).

79.3 Uniform Convergence e ERM

La consistenza dell'ERM richiede convergenza *uniforme* del rischio empirico su tutta la classe $\mathcal{H}$. Il teorema fondamentale (Vapnik-Chervonenkis):

TEOREMA FONDAMENTALE DI APPRENDIMENTO STATISTICO (PAC)

Sia $\mathcal{H}$ una classe con VC dimension $d_{\mathrm{VC}} < \infty$. Allora per ogni distribuzione $\mathcal{D}$ e ogni $\varepsilon, \delta > 0$, con campione di dimensione $N \geq N_0(\varepsilon, \delta, d_{\mathrm{VC}})$:

$$P\left(\sup_{h \in \mathcal{H}} |R(h) - \hat{R}_N(h)| > \varepsilon \right) \leq \delta$$

dove $N_0 = O\left(\frac{d_{\mathrm{VC}} + \ln(1/\delta)}{\varepsilon^2} \right)$.

Questa è condizione necessaria e sufficiente per la PAC-learnability: una classe è PAC-learnable se e solo se ha VC dimension finita.

PARTE 80

Landscape dell'Ottimizzazione nelle Reti Neurali Profonde

80.1 Classificazione dei Punti Critici

Per $f : \mathbb{R}^d \to \mathbb{R}$, un punto $\boldsymbol{\theta}^*$ è **critico** se $\nabla f(\boldsymbol{\theta}^*) = \mathbf{0}$. La classificazione tramite autovalori dell'Hessiana $\mathbf{H} = \nabla^2 f(\boldsymbol{\theta}^*)$:

Configurazione autovalori	Tipo di punto	Frequenza in reti profonde
Tutti $\lambda_i > 0$	Minimo locale	Rara per $d \gg 1$
Tutti $\lambda_i < 0$	Massimo locale	Estremamente rara
Mix positivi e negativi	Punto di sella	Dominante per $d \gg 1$
Alcuni $\lambda_i = 0$	Degenerato (flat direction)	Comune: simmetrie della rete

Argomento probabilistico

Se ogni autovalore è indipendentemente positivo con probabilità $p_+ < 1$, la probabilità che tutti gli d autovalori siano positivi è $(p_+)^d \to 0$ per $d \to \infty$. Per matrici aleatori GOE (Gaussian Orthogonal Ensemble), $p_+ \approx 1/2$ — quindi quasi nessun punto critico è un minimo locale per reti grandi. Questo è una buona notizia: i punti di sella sono più facili da sfuggire dei minimi locali.

80.2 Geometria dei Punti di Sella e Escape

Un punto di sella $\boldsymbol{\theta}^*$ ha direzioni di discesa (autovettori con $\lambda_i < 0$). Il gradiente sgd con rumore sfugge automaticamente ai punti di sella non-degenerati.

Per punti di sella **degenerati** (alcuni $\lambda_i = 0$), il GD può essere intrappolato. Il metodo **Perturbed Gradient Descent (PGD)** di Jin et al. (2017) aggiunge rumore periodicamente:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{\xi}_t - \alpha \nabla f(\boldsymbol{\theta}_t + \boldsymbol{\xi}_t), \quad \boldsymbol{\xi}_t = \begin{cases} \mathcal{U}(\mathcal{B}(0, r)) & \text{se } \|\nabla f\| \le \varepsilon,\ t \equiv 0 \pmod{T} \\ \mathbf{0} & \text{altrimenti} \end{cases}$$

TEOREMA (JIN ET AL., 2017)

Con learning rate $\alpha = O(\varepsilon / L)$ e perturbazione $r = O(\varepsilon^{1/4}/\ell)$ (con ℓ la costante di Lipschitz dell'Hessiana), PGD converge a un $(\varepsilon, \sqrt{\rho\varepsilon})$-SOSP (Second Order Stationary Point, dove $\|\nabla f\| \le \varepsilon$ e $\lambda_{\min}(\mathbf{H}) \ge -\sqrt{\rho\varepsilon}$) in $\tilde{O}(\varepsilon^{-2})$ iterazioni.

80.3 Flat Minima e Generalizzazione

Il fenomeno del **double descent** (Belkin et al., 2019) sfida la teoria bias-varianza classica. Con modelli overparametrizzati, l'errore di test cresce fino al punto di interpolazione ($N_{\text{params}} \approx N_{\text{data}}$) e poi decresce di nuovo. Perché?

I modelli overparametrizzati trovano la soluzione con norma minima (minimo con più simmetria e "piattezza") — questa ha generalizzazione migliore della soluzione overfit di modelli solo leggermente overparametrizzati. La misura di piattezza:

$$\text{Sharpness}(\boldsymbol{\theta}) = \max_{\|\boldsymbol{\epsilon}\| \le \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}) - \mathcal{L}(\boldsymbol{\theta})$$

L'ottimizzatore **SAM (Sharpness-Aware Minimization)** (Foret et al., 2021) minimizza esplicitamente la sharpness:

$$\min_{\boldsymbol{\theta}} \max_{\|\boldsymbol{\epsilon}\| \le \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}) \approx \min_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta} + \rho \frac{\nabla_{\boldsymbol{\theta}} \mathcal{L}}{\|\nabla_{\boldsymbol{\theta}} \mathcal{L}\|}\right)$$

Ogni step SAM richiede due forward pass: uno per trovare $\boldsymbol{\epsilon}^* = \rho \nabla / \|\nabla\|$ (perturbazione worst-case), e uno per calcolare il gradiente nel punto perturbato.

PARTE 81

Modelli Autoregressivi Avanzati: MADE, PixelCNN e VQ-VAE

81.1 MADE: Masked Autoencoder for Distribution Estimation

MADE (Germain et al., 2015) impone la struttura autoregressiva $p(\mathbf{x}) = \prod_{d=1}^{D} p(x_d | x_{1:d-1})$ tramite mascheratura delle connessioni dell'autoencoder. La chiave: si assegna a ogni neurone k nel layer l un ordine $m_k^l \in \{1, \ldots, D-1\}$. La maschera binaria della connessione tra layer $l-1$ e l:

$$M_{k,k'}^{l} = \mathbf{1}[m_k^l \geq m_{k'}^{l-1}]$$

Per la connessione di output (layer $L \to$ output d): $M_{d,k}^{\text{out}} = \mathbf{1}[d > m_k^L]$. Questo garantisce che l'output $\hat{x}_d$ non abbia nessun percorso computazionale che passi per neuroni che ricevono x_d come input — $\hat{x}_d$ dipende solo da $x_{1:d-1}$.

Order-agnostic training

Invece di fissare un ordinamento, MADE cambia l'ordinamento casualmente a ogni minibatch. L'autoencoder deve predire x_d in tutti gli ordinamenti possibili — impara una rappresentazione robusta all'ordine. Con un ordinamento casuale per ogni forward, il gradiente aggregato è:

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\sigma \sim S_D}[\nabla_\theta \mathcal{L}_\sigma] \approx \frac{1}{K} \sum_{k=1}^{K} \nabla_\theta \mathcal{L}_{\sigma_k}$$

In pratica $K = 1$ (un ordinamento per step) funziona bene. Per la stima della densità, la media su tutti gli ordinamenti fornisce una stima più accurata di $\log p(\mathbf{x})$.

81.2 PixelCNN: Convoluzione Causale 2D

PixelCNN (van den Oord et al., 2016) modella $p(\mathbf{x}) = \prod_{i=1}^{H} \prod_{j=1}^{W} p(x_{ij} | \mathbf{x}_{<(i,j)})$ usando convoluzioni causali 2D. Il pixel (i, j) nella scansione raster dipende da tutti i pixel con $(i'

Tipo A (per il primo layer, pixel corrente non incluso):

$$M^A_{r,c} = \begin{cases}1 & r
Tipo B (per layer successivi, pixel corrente incluso): come tipo A ma con $c \leq K/2$ invece di $c

PixelCNN++ (Salimans et al., 2017) sostituisce la distribuzione categorica su 256 livelli con la **discretized logistic mixture**:

$$P(x|\pi,\mu,s) = \sum_{k=1}^{K} \pi_k \left[\sigma\left(\frac{x+0.5-\mu_k}{s_k}\right) - \sigma\left(\frac{x-0.5-\mu_k}{s_k}\right)\right]$$

con $K = 5$ componenti logistiche. Questo permette di modellare la struttura continua dei valori pixel con molti meno parametri della categorica (3 parametri per componente vs 255).

81.3 VQ-VAE: Rappresentazioni Discrete con Codebook

VQ-VAE (van den Oord et al., 2017) introduce la quantizzazione vettoriale nel VAE per apprendere rappresentazioni discrete. Il codebook $\mathbf{E} = \{\mathbf{e}_k\}_{k=1}^{K} \subset \mathbb{R}^d$ è una matrice appresta. La codifica discreta:

$$z_q(\mathbf{x}) = \mathbf{e}_{k^*}, \quad k^* = \arg\min_k \|z_e(\mathbf{x}) - \mathbf{e}_k\|_2$$

Il gradiente non fluisce attraverso l'argmin (non differenziabile). Il **Straight-Through Estimator**: nel backward pass, si copia il gradiente di z_q direttamente su z_e come se fossero identici. Questo è un bias nel gradiente ma funziona in pratica perché $z_q \approx z_e$ a convergenza.

Loss completa e dynamics del codebook

$$\mathcal{L}_{\text{VQ}} = \underbrace{\|\mathbf{x} - D(z_q)\|^2}_{\text{ricostruzione}} + \underbrace{\|\text{sg}[z_e(\mathbf{x})] - z_q\|^2}_{\text{codebook update}} + \underbrace{\beta\|z_e(\mathbf{x}) - \text{sg}[z_q]\|^2}_{\text{commitment loss}}$$

Il termine codebook aggiorna $\mathbf{e}_{k^*}$ verso $z_e(\mathbf{x})$ (tramite gradient descent sul codebook). Il commitment loss ($\beta \approx 0.25$) spinge z_e verso il codebook, prevenendo che l'encoder "salti" da una entry all'altra.

Alternativa: EMA update del codebook

Invece del gradiente, si aggiorna il codebook con Exponential Moving Average — più stabile:

$$N_k \leftarrow \gamma N_k + (1-\gamma)\sum_i \mathbf{1}[k_i^* = k], \quad \mathbf{e}_k \leftarrow \gamma \mathbf{e}_k + (1-\gamma)\frac{\sum_i \mathbf{1}[k_i^*=k]\cdot z_e(\mathbf{x}^{(i)})}{N_k}$$

PARTE 82

Architetture per Sequenze Lunghissime: BigBird, S4, Hyena

82.1 BigBird: Attenzione Sparsa con Garanzie Teoriche

BigBird (Zaheer et al., 2020) dimostra che $O(n)$ connessioni di attenzione (invece di $O(n^2)$) sono sufficienti per l'universalità computazionale. Il pattern di attenzione usa tre componenti:

- **Random attention:** ogni query attende a r key random — connessioni $O(rn)$, r fisso
- **Window attention:** ogni query attende ai w token adiacenti (sliding window) — connessioni $O(wn)$
- **Global tokens:** g token speciali CLS-like attendono a tutti e vengono visti da tutti — connessioni $O(gn)$

TEOREMA (BIGBIRD — COMPLETEZZA DI TURING)

BigBird con combinazione dei tre pattern è Turing-completo e può simulare l'attenzione full tramite al più un numero costante di layer aggiuntivi. Formalmente, per ogni funzione computabile da un Transformer con attenzione full di profondità L, esiste un BigBird di profondità $L + O(1)$ che la computa.

La prova costruttiva usa i global token come "memoria di lavoro": (1) i global token raccolgono informazione da tutta la sequenza in un layer; (2) distribuiscono la risposta a tutti i token nel layer successivo. Due layer BigBird equivalgono a un layer di attenzione full.

82.2 S4: Structured State Space Model

S4 (Gu et al., 2022) risolve il problema dell'addestramento efficiente degli SSM con la parametrizzazione **HiPPO**. La matrice $\mathbf{A} \in \mathbb{R}^{N \times N}$ è progettata per comprimere

ottimalmente la storia — le sue colonne formano una base ortonormale di polinomi di Legendre:

$$A_{nk} = -\begin{cases}(2n+1)^{1/2}(2k+1)^{1/2} & n>k\ n+1 & n=k\ 0 & n
Con questa scelta, lo stato $\mathbf{h}(t)$ minimizza il MSE nella proiezione della storia $u(s),\ s \leq t$ sulla base di Legendre scalata — ottimalità nel senso della compressione.

Calcolo efficiente del kernel via NPLR

S4 diagonalizza la matrice $\bar{\mathbf{A}}$ tramite la decomposizione **Normal Plus Low Rank (NPLR)**: $\bar{\mathbf{A}} = \mathbf{V}\Lambda\mathbf{V}^* - \mathbf{p}\mathbf{q}^*$ dove $\mathbf{V}\Lambda\mathbf{V}^*$ è una matrice normale (diagonalizzabile) e $\mathbf{p}\mathbf{q}^*$ è un termine di rango 1.

Il kernel si calcola come integrale di Cauchy discreto:

$$\hat{K}_j = \mathbf{C}\bar{\mathbf{A}}^j\bar{\mathbf{B}} \Rightarrow \hat{K}(z) = \mathbf{C}(z\mathbf{I} - \bar{\mathbf{A}})^{-1}\bar{\mathbf{B}} = \text{sum of} \frac{c_k}{z - \lambda_k} \text{ via Woodbury}$$

La somma di risolventi $(z - \lambda_k)^{-1}$ si calcola in $O(N \log N)$ con FFT su $z = e^{2\pi i k/L}$, rendendo l'intero kernel $\bar{\mathbf{K}}$ computabile in $O(L \log L)$.

82.3 Hyena: Convoluzione Long-Range con Gating

Hyena (Poli et al., 2023) propone un'alternativa all'attenzione basata su convoluzioni implicite a lungo raggio e gating. L'operatore Hyena di ordine N su una sequenza $\mathbf{u} \in \mathbb{R}^{L \times d}$:

$$\mathbf{y} = (\mathbf{h}_N * \mathbf{x}_N) \odot (\mathbf{h}_{N-1} * (\mathbf{x}_{N-1} \odot (\cdots (\mathbf{h}_1 * \mathbf{x}_1) \odot \mathbf{x}_0) \cdots)))$$

dove $\mathbf{x}_k = \text{proj}_k(\mathbf{u})$ sono proiezioni dell'input, $\mathbf{h}_k$ sono filtri convoluzionali appresi impliciti (parametrizzati da una piccola rete), e $*$ è la convoluzione 1D (calcolata con FFT in $O(L \log L)$). L'intuizione: il gating (prodotto $\odot$) implementa una selezione contestuale, mentre le convoluzioni forniscono mixing a lungo raggio.

Confronto con l'Attenzione

La complessità di Hyena ordine-2 è $O(L \log L)$ vs $O(L^2 d)$ dell'attenzione. Per la qualità: Hyena non ha content-based addressing (le convoluzioni sono data-independent, a differenza del prodotto QK che varia per ogni coppia di token). Su sequenze molto lunghe ($L > 8K$), Hyena supera l'attenzione troncata per task che richiedono correlazioni a lungo raggio.

PARTE 83

Teoria del Transfer Learning e Domain Adaptation

83.1 Bound di Generalizzazione con Domain Shift

Nella situazione di transfer learning, si ha una distribuzione sorgente $\mathcal{D}_S$ e una target $\mathcal{D}_T$ diverse. Ben-David et al. (2010) dimostrano un bound fondamentale:

TEOREMA (BEN-DAVID ET AL., 2010)

Per ogni ipotesi $h \in \mathcal{H}$, l'errore sul dominio target è limitato da:

$$R_T(h) \leq R_S(h) + \frac{1}{2}\hat{d}_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda^*$$

dove $\hat{d}_{\mathcal{H}}$ è l'$\mathcal{H}$-divergenza empirica e $\lambda^* = \min_{h' \in \mathcal{H}}[R_S(h') + R_T(h')]$ è il rischio combinato minimo ottimale.

La $\mathcal{H}$-divergenza misura la discrepanza tra le due distribuzioni "vista" dalla classe di ipotesi:

$$d_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) = 2 \sup_{h \in \mathcal{H}} |P_S[h(\mathbf{x}) = 1] - P_T[h(\mathbf{x}) = 1]|$$

Si stima addestrando un classificatore binario a distinguere $\mathcal{D}_S$ da $\mathcal{D}_T$: $\hat{d}_{\mathcal{H}} = 2(1 - 2\varepsilon_d)$ dove ε_d è il suo errore. Se le due distribuzioni sono indistinguibili ($\varepsilon_d \approx 0.5$), il transfer è facilitato ($\hat{d}_{\mathcal{H}} \approx 0$).

83.2 DANN: Domain-Adversarial Neural Network

DANN (Ganin et al., 2016) minimizza $d_{\mathcal{H}}$ con un approccio avversariale. L'architettura ha tre componenti: feature extractor G_f, label predictor G_y, e domain discriminator G_d. Il **Gradient Reversal Layer (GRL)** è un layer identità nel forward ma moltiplica i gradienti per $-\lambda$ nel backward:

$$\text{Forward: } R_\lambda(\mathbf{x}) = \mathbf{x}$$

$$\text{Backward: } \frac{dR_\lambda}{d\mathbf{x}} = -\lambda \mathbf{I}$$

La loss combinata:

$$\mathcal{L}(\boldsymbol{\theta}_f, \boldsymbol{\theta}_y, \boldsymbol{\theta}_d) = \mathcal{L}_y(G_y(G_f(\mathbf{x}_S)), y_S) - \lambda\, \mathcal{L}_d(G_d(G_f(\mathbf{x})), d)$$

Minimizzare rispetto a $\boldsymbol{\theta}_f, \boldsymbol{\theta}_y$, massimizzare rispetto a $\boldsymbol{\theta}_d$. Il GRL trasforma il max in min per $\boldsymbol{\theta}_f$ — il feature extractor viene addestrato a rendere i domini indistinguibili per il discriminatore.

Punto fisso dell'ottimizzazione

All'equilibrio di Nash, il discriminatore di dominio non riesce a distinguere i due domini ($\varepsilon_d = 0.5$), quindi $\hat{d}_{\mathcal{H}} = 0$ — il bound di Ben-David diventa $R_T(h) \leq R_S(h) + \lambda^*$. Il trasferimento è ottimale quando λ^* è piccolo — entrambi i task sono risolubili con la stessa ipotesi.

83.3 Maximum Mean Discrepancy

La **Maximum Mean Discrepancy (MMD)** è un'alternativa all'$\mathcal{H}$-divergenza. Data una funzione kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (RKHS), la MMD misura la differenza tra embedding della distribuzione:

$$\text{MMD}^2(\mathcal{D}_S, \mathcal{D}_T) = \mathbb{E}_{x,x' \sim \mathcal{D}_S}[k(x,x')] - 2\mathbb{E}_{x \sim \mathcal{D}_S, y \sim \mathcal{D}_T}[k(x,y)] + \mathbb{E}_{y,y' \sim \mathcal{D}_T}[k(y,$$

Con kernel Gaussiano $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2/\sigma^2)$, la MMD è zero se e solo se le due distribuzioni sono identiche. La stima empirica su campioni $\{x_i\}_{i=1}^m$ e $\{y_j\}_{j=1}^n$:

$$\widehat{\text{MMD}}^2 = \frac{1}{m^2}\sum_{i,i'} k(x_i, x_{i'}) - \frac{2}{mn}\sum_{i,j} k(x_i, y_j) + \frac{1}{n^2}\sum_{j,j'} k(y_j, y_{j'})$$

La MMD può essere usata come loss di domain alignment: minimizzare $\widehat{\text{MMD}}^2$ tra le rappresentazioni delle due distribuzioni (DAN — Deep Adaptation Networks).

PARTE 84

Mixture Density Networks e Previsione Probabilistica

Come produrre stime di incertezza calibrate: MDN, conformal prediction e deep ensembles.

84.1 Mixture Density Networks

Le MDN (Bishop, 1994) modellano la distribuzione condizionale $p(y|\mathbf{x})$ come miscela di gaussiane, catturando la multimodalità che una rete standard con output scalare non può rappresentare:

$$p(y|\mathbf{x}) = \sum_{k=1}^{K} \pi_k(\mathbf{x})\, \mathcal{N}(y; \mu_k(\mathbf{x}), \sigma_k^2(\mathbf{x}))$$

La rete produce i parametri della miscela con attivazioni specifiche:

$$\boldsymbol{\pi}(\mathbf{x}) = \mathrm{softmax}(\mathbf{z}^{\pi}), \quad \boldsymbol{\mu}(\mathbf{x}) = \mathbf{z}^{\mu} \text{ (lineare)}, \quad \boldsymbol{\sigma}(\mathbf{x}) = \exp(\mathbf{z}^{\sigma}) \text{ (positività)}$$

Training con negative log-likelihood (NLL):

$$\mathcal{L}_{\text{MDN}} = -\sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k(\mathbf{x}^{(i)})\, \mathcal{N}(y^{(i)}; \mu_k(\mathbf{x}^{(i)}), \sigma_k^2(\mathbf{x}^{(i)}))$$

Numericamente stabile con log-sum-exp: $\log \sum_k \pi_k \mathcal{N}_k = \mathrm{lsexp}(\log \pi_k + \log \mathcal{N}_k)$.

Gradiente rispetto ai parametri della miscela

Definiamo $r_k^{(i)} = \pi_k \mathcal{N}(y^{(i)}; \mu_k, \sigma_k^2) / \sum_j \pi_j \mathcal{N}_j$ (responsabilità). I gradienti hanno forme semplici:

$$\frac{\partial \mathcal{L}}{\partial \mu_k} = \sum_i r_k^{(i)} \frac{\mu_k(\mathbf{x}^{(i)}) - y^{(i)}}{\sigma_k^2(\mathbf{x}^{(i)})}$$

$$\frac{\partial \mathcal{L}}{\partial \sigma_k} = \sum_i r_k^{(i)} \left[\frac{1}{\sigma_k} - \frac{(y^{(i)} - \mu_k)^2}{\sigma_k^3} \right]$$

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \sum_i (r_k^{(i)} - \pi_k) \quad \text{(attraverso softmax)}$$

Il gradiente è l'analogo dell'E-M: le responsabilità r_k pesano il contributo di ogni componente, come nell'EM per GMM.

84.2 Conformal Prediction: Garanzie Teoriche Senza Assunzioni Distribuzionali

La conformal prediction (Vovk et al., 1999) produce prediction set con copertura garantita marginalmente sotto la sola assunzione di scambiabilità (più debole di i.i.d.).

ALGORITMO — SPLIT CONFORMAL PREDICTION

1. Dividi i dati in training set $\mathcal{D}_{\text{train}}$ e calibration set $\mathcal{D}_{\text{cal}}$
2. Addestra il modello $\hat{f}$ su $\mathcal{D}_{\text{train}}$
3. Calcola nonconformity scores $s_i = s(\mathbf{x}_i, y_i, \hat{f})$ su $\mathcal{D}_{\text{cal}}$ (es. $s_i = |y_i - \hat{f}(\mathbf{x}_i)|$ per regressione)
4. Per il livello $1-\alpha$: $\hat{q} = \text{Quantile}(s_1, \ldots, s_n; \lceil (n+1)(1-\alpha) \rceil / n)$
5. Prediction set: $C(\mathbf{x}_{n+1}) = \{y : s(\mathbf{x}_{n+1}, y, \hat{f}) \leq \hat{q}\}$

TEOREMA (COBERTURA MARGINAL GARANTITA)

Se $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n+1}, y_{n+1})$ sono scambiabili:

$$1 - \alpha \leq P(y_{n+1} \in C(\mathbf{x}_{n+1})) \leq 1 - \alpha + \frac{1}{n+1}$$

La garanzia è esatta, senza ipotesi sulla distribuzione dei dati o sul modello $\hat{f}$.

La prova usa la scambiabilità: i $n+1$ score $s_1, \ldots, s_n, s_{n+1}$ sono scambiabili, quindi $P(s_{n+1} \leq \hat{q}_\alpha) \geq 1-\alpha$ dove $\hat{q}_\alpha$ è il $\lceil (n+1)(1-\alpha) \rceil$-esimo quantile degli score di calibrazione.

84.3 Deep Ensembles e Stima dell'Incertezza

I Deep Ensembles (Lakshminarayanan et al., 2017) combinano M reti neurali con inizializzazioni diverse per stimare l'incertezza epistémica:

$$p(y|\mathbf{x}, \mathcal{D}) \approx \frac{1}{M}\sum_{m=1}^{M} p_{\theta_m}(y|\mathbf{x})$$

$$\mathbb{E}[y|\mathbf{x}] \approx \bar{\mu} = \frac{1}{M}\sum_{m} \mu_{\theta_m}(\mathbf{x})$$

$$\mathrm{Var}[y|\mathbf{x}] \approx \frac{1}{M}\sum_{m} \sigma^2_{\theta_m}(\mathbf{x}) + \frac{1}{M}\sum_{m} (\mu_{\theta_m}(\mathbf{x}) - \bar{\mu})^2$$

Il primo termine è l'incertezza *aleatoria* (rumore intrinseco dei dati), il secondo è l'incertezza *epistemica* (disaccordo tra modelli). Su distribuzioni fuori dalla distribuzione di training (OOD), l'incertezza epistemica è alta — i modelli dissentono.

PARTE 85

Loss Functions Avanzate: Detection, Segmentazione e Ranking

85.1 IoU, GIoU, DIoU, CIoU per Object Detection

Per la regressione delle bounding box, le loss basate su IoU sono superiori alle loss L_2 sui parametri perché invarianti alla scala.

IoU Loss: $\mathcal{L}_{\text{IoU}} = 1 - \frac{|B_p \cap B_g|}{|B_p \cup B_g|}$. Problema: gradiente zero per box non-sovrapposte.

GIoU (Generalized IoU, Rezatofighi et al., 2019): aggiunge la penalità per la box minima contenente entrambe C:

$$\mathcal{L}_{\text{GIoU}} = 1 - \text{IoU} + \frac{|C \setminus (B_p \cup B_g)|}{|C|} \in [0, 2]$$

DIoU (Distance IoU): penalizza la distanza tra centri $\rho(\mathbf{b}_p, \mathbf{b}_g)$ normalizzata per la diagonale c del minimo box:

$$\mathcal{L}_{\text{DIoU}} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}_p, \mathbf{b}_g)}{c^2}$$

CIoU (Complete IoU): aggiunge penalità sul rapporto d'aspetto tramite la variabile v e α:

$$v = \frac{4}{\pi^2}\left(\arctan\frac{w_g}{h_g} - \arctan\frac{w_p}{h_p}\right)^2, \quad \alpha = \frac{v}{1 - \text{IoU} + v}$$

$$\mathcal{L}_{\text{CIoU}} = \mathcal{L}_{\text{DIoU}} + \alpha v$$

Il gradiente di CIoU rispetto a (x_p, y_p, w_p, h_p) è calcolabile per backpropagation grazie alla differenziabilità di tutte le componenti. Il termine α è trattato come costante (stop gradient) per stabilità.

85.2 Dice Loss, Tversky e Focal-Tversky

Per segmentazione semantica con classi sbilanciate, la cross-entropy dà peso eccessivo alla classe dominante.

Il coefficiente di **Sørensen-Dice** per due insiemi binari è:

$$\mathrm{DSC} = \frac{2|P \cap G|}{|P| + |G|}$$

La **Dice Loss** per predizioni soft $p_i \in [0,1]$ e label $g_i \in \{0,1\}$:

$$\mathcal{L}_{\mathrm{Dice}} = 1 - \frac{2\sum_i p_i g_i + \varepsilon}{\sum_i p_i + \sum_i g_i + \varepsilon}$$

Il gradiente rispetto a p_i:

$$\frac{\partial \mathcal{L}_{\mathrm{Dice}}}{\partial p_i} = -\frac{2g_i(\sum_j p_j + \sum_j g_j) - (2\sum_j p_j g_j + \varepsilon)}{(\sum_j p_j + \sum_j g_j + \varepsilon)^2}$$

La **Tversky Loss** (Salehi et al., 2017) generalizza Dice con pesi asimmetrici:

$$\mathrm{TI}(\alpha,\beta) = \frac{\sum_i p_i g_i}{\sum_i p_i g_i + \alpha\sum_i p_i(1-g_i) + \beta\sum_i (1-p_i)g_i}$$

$$\mathcal{L}_{\mathrm{Tversky}} = 1 - \mathrm{TI}(\alpha,\beta)$$

Per $\alpha < \beta$ si penalizzano di più i falsi negativi (utile per lesioni piccole: meglio falso allarme che lesione persa). La **Focal-Tversky** aggiunge il focusing per esempi difficili: $\mathcal{L}_{\mathrm{FTL}} = (1 - \mathrm{TI})^{\gamma}$.

85.3 Learning to Rank: RankNet, LambdaRank, LambdaMART

Nel Learning to Rank, l'obiettivo è ordinare documenti per rilevanza rispetto a una query.

RankNet (Burges et al., 2005): modella $P_{ij} = \sigma(s_i - s_j)$ e minimizza cross-entropy:

$$\mathcal{L}_{\text{RN}} = -\sum_{(i,j):i \succ j} \left[\bar{P}_{ij} \log \sigma(s_i - s_j) + (1 - \bar{P}_{ij}) \log(1 - \sigma(s_i - s_j)) \right]$$

Il gradiente ha la forma di "forze" tra coppie: $\lambda_{ij} = \partial \mathcal{L} / \partial s_i = \sigma(s_i - s_j) - 1$ per $j \prec i$.

LambdaRank: scala i gradienti λ_{ij} per $|\Delta \text{NDCG}_{ij}|$ — la variazione di NDCG se si scambiassero i e j nel ranking:

$$\lambda_{ij}^{\text{LR}} = |\Delta \text{NDCG}_{ij}| \cdot \frac{-1}{1 + e^{s_i - s_j}}$$

Questo ottimizza direttamente NDCG in modo approssimato. **LambdaMART** combina LambdaRank con gradient boosting di alberi decisionali — lo stato dell'arte per ranking Web.

NDCG: Normalized Discounted Cumulative Gain

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log_2(i+1)}, \qquad \text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}$$

dove r_i è il grado di rilevanza del documento in posizione i e IDCG è il DCG del ranking ideale (perfetto). NDCG è non-differenziabile per la presenza dell'argmax nel ranking.

PARTE 86

Reti Neurali per Dati Tabulari

86.1 TabNet: Selezione Sequenziale di Feature

TabNet (Arik e Pfister, 2021) usa un'attention sparsa sequenziale per selezionare le feature rilevanti ad ogni step. L'architettura ha N_{steps} passi, ognuno dei quali seleziona un sottoinsieme di feature:

$$\mathbf{M}_i[b] = \text{sparsemax}(\mathbf{P}_{i-1}[b] \cdot h_{\text{att}}(\mathbf{a}^{[i]}[b]))$$
$$\mathbf{P}_i[b] = \mathbf{P}_{i-1}[b] \odot (\eta - \mathbf{M}_i[b])$$
$$h_i[b] = f_i(\mathbf{M}_i[b] \odot \mathbf{a}[b])$$

dove $\mathbf{a} \in \mathbb{R}^{B \times d}$ sono le feature di input, $\mathbf{M}_i \in [0,1]^{B \times d}$ è la maschera di attenzione, $\mathbf{P}_i$ è la penalità di utilizzo (incoraggia ogni feature ad essere selezionata al più una volta), e h_{att} è un trasformatore di attenzione condiviso.

Sparsemax: definizione e calcolo

Sparsemax (Martins e Fernandez, 2016) proietta sul simplesso con norma minima:

$$\text{sparsemax}(\mathbf{z}) = \arg\min_{\mathbf{p} \in \Delta^{K-1}} \|\mathbf{p} - \mathbf{z}\|_2^2$$

La soluzione è: $p_i^* = [z_i - \tau(\mathbf{z})]_+ = \max(0, z_i - \tau(\mathbf{z}))$ dove $\tau(\mathbf{z}) = (\sum_{j \in S(\mathbf{z})} z_j - 1)/|S(\mathbf{z})|$ e $S(\mathbf{z}) = \{j : z_j > \tau(\mathbf{z})\}$. La soglia τ si trova in $O(K \log K)$ ordinando gli elementi di $\mathbf{z}$.

Loss di entropia sparsa per regolarizzazione

TabNet aggiunge una loss di entropia per incoraggiare maschere nette:

$$\mathcal{L}_{\text{sparse}} = -\frac{1}{N_{\text{steps}} \cdot B} \sum_i \sum_b \sum_j M_i[b,j] \log(M_i[b,j] + \varepsilon)$$

Questa loss penalizza le distribuzioni uniformi (alta entropia) — incoraggia il modello a selezionare poche feature con alta confidenza.

86.2 SAINT: Self-Attention su Righe e Colonne

SAINT (Somepalli et al., 2021) usa il Transformer sia sulle feature (colonne) che sugli esempi (righe) per catturare interazioni bidimensionali nei dati tabulari.

Ogni esempio $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_d^{(i)})$ viene embedding per feature: $\mathbf{e}_j^{(i)} = E_j(x_j^{(i)}) \in \mathbb{R}^h$. L'attenzione sulle colonne (inter-feature) calcola interazioni tra feature dello stesso esempio:

$$\text{ColumnAttn}: \mathbf{E}^{(i)} = \text{TransformerLayer}(\mathbf{e}_1^{(i)}, \ldots, \mathbf{e}_d^{(i)})$$

L'attenzione sulle righe (inter-sample) calcola interazioni tra esempi diversi per la stessa feature:

$$\text{RowAttn}_j: [e_j^{(1)}, \ldots, e_j^{(B)}] = \text{TransformerLayer}([e_j^{(1)}, \ldots, e_j^{(B)}])$$

L'alternanza row/column attention permette al modello di catturare sia le interazioni tra feature che le similarità tra esempi, portando a generalizzazione superiore rispetto ai metodi puramente per feature.

PARTE 87

Position Encoding Avanzato: ALiBi, YaRN e Contesto Ultra-Lungo

87.1 ALiBi: Attention with Linear Biases

ALiBi (Press et al., 2022) non aggiunge embedding di posizione agli input — incorpora la posizione direttamente nei punteggi di attenzione come un bias lineare dipendente dalla distanza:

$$A_{ij} = \frac{\mathbf{q}_i^T \mathbf{k}_j}{\sqrt{d_k}} - m \cdot |i - j|$$

dove m è uno scalare fisso (non appreso) che varia per testa. Con H teste, la testa h usa:

$$m_h = 2^{-8h/H}$$

Per $H = 8$: $m_1 = 1, m_2 = 1/2, m_3 = 1/4, \ldots, m_8 = 1/128$. Alcune teste hanno penalità forte (attenzione locale) e altre debole (attenzione lunga distanza).

Proprietà di extrapolazione

Il vantaggio chiave di ALiBi: la penalità lineare è monotona nella distanza — il modello non ha mai visto distanze negative o zero nel training, ma la funzione lineare extrapola naturalmente a distanze arbitrariamente grandi. A differenza del sinusoidale o del RoPE (che possono produrre pattern periodici a distanze insolite), ALiBi mantiene la penalità crescente in modo prevedibile.

87.2 YaRN: Estensione del Contesto senza Ri-training Completo

YaRN (Peng et al., 2023) estende la lunghezza di contesto di modelli RoPE da L_{orig} a L_{new} con minimo fine-tuning. La chiave: trattare diversamente le dimensioni ad alta e bassa frequenza.

Per la dimensione i, la lunghezza d'onda di RoPE è $\lambda_i = 2\pi/\theta_i = 2\pi \cdot 10000^{2i/d}$. Le dimensioni con $\lambda_i < L_{\text{orig}}$ (alta frequenza, ciclo completo nel contesto di training) vengono interpolate; quelle con $\lambda_i \geq L_{\text{orig}}$ (bassa frequenza, periodo lungo) vengono estrapolate:

$$\theta_i^{\text{new}} = \begin{cases} \theta_i & \lambda_i < 1 \text{ (già abbastanza alta frequenza)} \\ \theta_i / s & 1 \leq \lambda_i < \beta \text{ (interpolazione lineare)} \\ \frac{\theta_i}{\gamma \cdot s} & \lambda_i \geq \beta \text{ (estrapolazione)} \end{cases}$$

dove $s = L_{\text{new}}/L_{\text{orig}}$, $\beta = 32$ e γ è un fattore di ottimizzazione. Questa strategia non uniforme riduce la perplexity del 15-30% rispetto all'interpolazione lineare semplice (PI).

87.3 NTK-aware Interpolation e LongRoPE

L'interpolazione NTK-aware (bloc1ntu, 2023) scala la base di RoPE invece delle frequenze: $\theta_i^{\text{new}} = \theta_{\text{base}}^{\text{new}\,2i/d}$ con $\theta_{\text{base}}^{\text{new}} = \theta_{\text{base}} \cdot s^{d/(d-2)}$. Questo distribuisce la modifica uniformemente su tutte le dimensioni senza richiedere di identificare la soglia β.

LongRoPE (Ding et al., 2024) fa un passo ulteriore: cerca i fattori di scala ottimali $\{s_i^*\}$ per ogni dimensione tramite ricerca evolutiva su un mini corpus di documenti lunghi, minimizzando la perplexity. Il training finale usa:

$$\theta_i^* = \theta_{\text{base}}^{2i/d} \cdot s_i^*, \quad \{s_i^*\} = \arg\min_{\{s_i\}} \mathcal{L}_{\text{val}}\Big(\text{modello con } \{\theta_i = \theta_{\text{base}}^{2i/d}/s_i\}\Big)$$

LongRoPE raggiunge contesti di 2M token con solo 1000 step di fine-tuning — la ricerca dei fattori ottimali è il vero contributo algoritmico.

PARTE 88

Ottimizzazione delle Preferenze: DPO, KTO, IPO e SimPO

88.1 Richiamo di DPO e sue Proprietà

La loss DPO (derivata nella Parte 36 del Volume I) minimizza:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$

Il gradiente di DPO rispetto a π_θ:

$$\nabla_{\pi_\theta}\mathcal{L}_{\text{DPO}} = -\beta\,\mathbb{E}\big[\sigma(\hat{r}_l - \hat{r}_w)(\nabla_{\pi_\theta}\log\pi_\theta(y_w|x) - \nabla_{\pi_\theta}\log\pi_\theta(y_l|x))\big]$$

dove $\hat{r} = \beta\log(\pi_\theta/\pi_{\text{ref}})$ è il reward implicito. Il peso $\sigma(\hat{r}_l - \hat{r}_w) \in (0,1)$: quando la preferenza è già ben imparata ($\hat{r}_w \gg \hat{r}_l$), il peso è piccolo — DPO impara di più dagli esempi "difficili".

88.2 KTO: Kahneman-Tversky Optimization

KTO (Ethayarajh et al., 2024) si ispira alla Prospect Theory (Kahneman e Tversky, 1979): le persone sono avverse alle perdite asimmetricamente — la perdita di 10€ fa più "male" del guadagno di 10€ fa "bene". A differenza di DPO, KTO funziona su singole risposte con label binaria:

$$\mathcal{L}_{\text{KTO}}(\pi_\theta;\pi_{\text{ref}}) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[w(y)(1 - v_{\text{KTO}}(x,y;\beta))]$$

$$v_{\text{KTO}}(x,y;\beta) = \begin{cases} \sigma\Big(\beta(\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} - z_{\text{ref}})\Big) & y \in \mathcal{D}_{\text{desider.}} \\ \sigma\Big(-\beta(\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} - z_{\text{ref}})\Big) & y \in \mathcal{D}_{\text{indesidr.}} \end{cases}$$

dove $z_{\text{ref}} = \mathbb{E}_{(x',y')\sim\mathcal{D}}[\beta \text{KL}(\pi_\theta(\cdot|x')\|\pi_{\text{ref}}(\cdot|x'))]$ è il punto di riferimento (KL atteso) e $w(y) \in \{w_D, w_U\}$ con $w_D, w_U > 0$ bilanciano l'asimmetria. Per $w_D = 1, w_U = 1$: effetto simmetrico; per $w_D

88.3 IPO: Identity Preference Optimization

IPO (Azar et al., 2024) mostra che DPO soffre di un problema di ottimizzazione eccessiva: per coppie con forte preferenza, $\log(\pi/\pi_{\text{ref}}) \to \pm\infty$ — il modello si allontana indefinitamente dalla policy di riferimento. IPO aggiunge una regolarizzazione quadratica implicita:

$$\mathcal{L}_{\text{IPO}} = \mathbb{E}\left[\left(h_{\pi_\theta}(x, y_w, y_l) - \frac{1}{2\beta}\right)^2\right]$$

$$h_{\pi_\theta}(x, y_w, y_l) = \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}$$

La soluzione ottimale di IPO soddisfa $h_{\pi^*} = 1/(2\beta)$ — il gap nei log-ratio è fisso, proporzionale a $1/\beta$. Per β grande: la policy rimane vicina alla reference (conservativa); per β piccolo: la policy può allontanarsi di più.

88.4 SimPO: Senza Reference Policy

SimPO (Meng et al., 2024) elimina completamente la policy di riferimento, usando come reward la log-likelihood media per token:

$$r_{\text{SimPO}}(x, y) = \frac{\beta}{|y|} \log \pi_\theta(y|x)$$

La loss SimPO con margine $\gamma > 0$:

$$\mathcal{L}_{\text{SimPO}} = -\mathbb{E}\left[\log \sigma\left(\frac{\beta}{|y_w|}\log \pi_\theta(y_w|x) - \frac{\beta}{|y_l|}\log \pi_\theta(y_l|x) - \gamma\right)\right]$$

Il margine γ è essenziale: senza reference policy, senza margine la loss è soddisfatta banalmente da $\pi_\theta = \pi_{\text{ref}}$. La normalizzazione per lunghezza $|y|$ previene la preferenza per risposte brevi (che hanno log-likelihood più alta per lunghezza).

Confronto tra i metodi di alignment

Metodo	Usa ref. policy	Tipo di feedback	Loss
DPO	Sì	Coppie (y_w, y_l)	Cross-entropy su preferenze
KTO	Sì	Singole risposte + label	Prospect Theory
IPO	Sì	Coppie	Loss quadratica
SimPO	No	Coppie	Cross-entropy con margine
RLHF-PPO	Sì	Reward model	PPO con KL penalty

PARTE 89

Normalizing Flows Avanzati: Glow, FFJORD e Neural Spline Flows

89.1 Glow: Convoluzione Invertibile 1×1

Glow (Kingma e Dhariwal, 2018) introduce la **convoluzione invertibile 1×1** come layer generativo per immagini. Per un tensore di feature $\mathbf{h} \in \mathbb{R}^{H \times W \times C}$, una convoluzione 1×1 con peso $\mathbf{W} \in \mathbb{R}^{C \times C}$ mescola i canali:

$$\mathbf{y}_{hw} = \mathbf{W}\mathbf{h}_{hw}, \qquad \log|\det \mathbf{J}| = H \cdot W \cdot \log|\det \mathbf{W}|$$

L'inversione: $\mathbf{h}_{hw} = \mathbf{W}^{-1}\mathbf{y}_{hw}$. Per eficienza, si usa la decomposizione LU: $\mathbf{W} = \mathbf{PLU}$ con $\mathbf{P}$ una matrice di permutazione (fissa), $\mathbf{L}$ triangolare inferiore con diagonale 1, $\mathbf{U}$ triangolare superiore. Il log-determinante:

$$\log|\det \mathbf{W}| = \sum_{i} \log|U_{ii}|$$

solo dal prodotto degli elementi diagonali di $\mathbf{U}$ — $O(C)$ invece di $O(C^3)$ per il determinante diretto.

ActNorm: normalizzazione inizializzata dai dati

Glow usa ActNorm invece di Batch Norm per la compatibilità con batch size piccoli. Scala e bias per canale, inizializzati tramite un data-dependent initialization (primo batch):

$$s_c = 1/\text{std}(\mathbf{h}_{:,:,c}), \quad b_c = -\text{mean}(\mathbf{h}_{:,:,c}) \cdot s_c$$
$$\text{ActNorm}(\mathbf{h})_{hwc} = s_c \cdot h_{hwc} + b_c, \quad \log|\det \mathbf{J}| = H \cdot W \sum_c \log|s_c|$$

89.2 FFJORD: Continuous Normalizing Flows

FFJORD (Grathwohl et al., 2018) generalizza i normalizing flows a trasformazioni continue tramite ODE. La trasformazione $\mathbf{z}(t_0) \to \mathbf{z}(t_1)$ segue:

$$\frac{d\mathbf{z}}{dt} = f_\theta(\mathbf{z}(t), t)$$

Il cambio di log-density è dato dall'equazione di Fokker-Planck (Liouville):

$$\frac{d \log p(\mathbf{z}(t))}{dt} = -\mathrm{tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right)$$

Lo stimatore di Hutchinson calcola la traccia in $O(d)$ invece di $O(d^2)$:

$$\mathrm{tr}\left(\frac{\partial f}{\partial \mathbf{z}}\right) \approx \boldsymbol{\varepsilon}^T \frac{\partial f}{\partial \mathbf{z}} \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

La log-likelihood è calcolata risolvendo l'ODE aumentata per $(\mathbf{z}(t), \log p(\mathbf{z}(t)))$ simultaneamente, usando un solutore ODE adattivo.

89.3 Neural Spline Flows

I Neural Spline Flows (Durkan et al., 2019) sostituiscono la trasformazione affine dei coupling layers con spline razionale-quadratiche monotone — molto più espressive con un numero simile di parametri.

Una **spline razionale-quadratica** su $[x_L, x_R]$ è definita da $K+1$ nodi $(x^{(k)}, y^{(k)})$ e K derivate $d^{(k)} > 0$. In ogni segmento $(x^{(k)}, x^{(k+1)})$:

$$g(x;\theta) = \frac{(y^{(k+1)} - y^{(k)})\xi^2 + (d^{(k)} + d^{(k+1)} - 2s_k)\xi(1-\xi) + d^{(k)}\xi}{\frac{d^{(k)}+d^{(k+1)}-2s_k}{s_k}\xi(1-\xi) + \frac{d^{(k)}}{s_k}(1-\xi) + \frac{d^{(k+1)}}{s_k}\xi}$$

dove $\xi = (x - x^{(k)})/(x^{(k+1)} - x^{(k)}) \in [0,1]$ e
$s_k = (y^{(k+1)} - y^{(k)})/(x^{(k+1)} - x^{(k)})$ è la pendenza del segmento. La derivata è analitica e strettamente positiva — la funzione è monotona crescente e invertibile.

PARTE 90

Momentum, Accelerazione e Geometria dell'Ottimizzazione

90.1 Heavy Ball: Analisi per Funzioni Quadratiche

Per funzioni quadratiche $f(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^T\mathbf{A}\boldsymbol{\theta} - \mathbf{b}^T\boldsymbol{\theta}$ (con $\mathbf{A}$ SPD, autovalori $0 < \mu \leq \lambda \leq L$), il Heavy Ball:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha\mathbf{A}(\boldsymbol{\theta}_t - \boldsymbol{\theta}^*) + \beta(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1})$$

è equivalente al sistema lineare $\boldsymbol{\epsilon}_{t+1} = (1+\beta)\boldsymbol{\epsilon}_t - \beta\boldsymbol{\epsilon}_{t-1} - \alpha\mathbf{A}\boldsymbol{\epsilon}_t$ con $\boldsymbol{\epsilon}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}^*$. Nella base degli autovettori di $\mathbf{A}$, la k-esima coordinata segue:

$$\epsilon_{t+1,k} = (1 + \beta - \alpha\lambda_k)\epsilon_{t,k} - \beta\epsilon_{t-1,k}$$

Il polinomio caratteristico è $r^2 - (1+\beta-\alpha\lambda_k)r + \beta = 0$. Le radici devono avere modulo < 1 per convergenza. La scelta ottimale minimizza il raggio spettrale:

$$\alpha^* = \frac{4}{(\sqrt{L}+\sqrt{\mu})^2}, \quad \beta^* = \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2$$

Con radici $r_{1,2}^* = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$, il raggio spettrale è $\rho^* = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ — il tasso di convergenza è $O(\rho^{*2t})$, quadraticamente migliore rispetto a GD $(\kappa-1)/(\kappa+1))^t$.

90.2 Nesterov Accelerated Gradient: Dimostrazione Completa

NAG usa il gradiente valutato nel "look-ahead" $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}_t + \frac{t-1}{t+2}(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1})$. La dimostrazione della convergenza ottimale usa la funzione di Lyapunov:

1. Definiamo $\Phi_t = t^2(f(\boldsymbol{\theta}_t) - f^*) + L\|\mathbf{v}_t - \boldsymbol{\theta}^*\|^2$ con $\mathbf{v}_t = t\boldsymbol{\theta}_t - (t-1)\boldsymbol{\theta}_{t-1}$
   Scelta della funzione di Lyapunov (non ovvia — si trova per tentativi)
2. Si dimostra che $\Phi_{t+1} \leq \Phi_t$ per ogni t (non aumenta mai)
   Calcolo algebrico usando la L-smoothness di f e la definizione di NAG
3. Quindi $T^2(f(\boldsymbol{\theta}_T) - f^*) \leq \Phi_T \leq \Phi_1 = L\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2$
   Applicando la monotonia e la non-negatività di $f(\boldsymbol{\theta}_t) - f^*$
4. $f(\boldsymbol{\theta}_T) - f^* \leq \dfrac{L\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2}{T^2}$
   Dividendo per T^2

$$f(\boldsymbol{\theta}_T) - f^* \leq \frac{2L\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2}{(T+1)^2} = O(1/T^2)$$

Per il GD standard: $O(1/T)$. NAG è ottimale — il lower bound di oracle complexity per metodi del primo ordine su funzioni convesse L-smooth è $\Omega(1/T^2)$.

90.3 SAM: Sharpness-Aware Minimization

SAM (Foret et al., 2021) mira a trovare minima piatti (flat minima) che generalizzano meglio. L'obiettivo:

$$\min_{\boldsymbol{\theta}} \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$$

Il massimo interno ha soluzione approssimata: $\boldsymbol{\epsilon}^*(\boldsymbol{\theta}) \approx \rho \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) / \|\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})\|_2$. Il gradiente dell'obiettivo esterno:

$$\nabla_{\boldsymbol{\theta}} \left[\max_{\|\boldsymbol{\epsilon}\| \leq \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}) \right] \approx \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}^*)|_{\boldsymbol{\theta} + \boldsymbol{\epsilon}^*}$$

In pratica SAM richiede due forward+backward per step: il primo calcola $\boldsymbol{\epsilon}^*$, il secondo calcola il gradiente nel punto perturbato. Il costo computazionale raddoppia, ma la generalizzazione migliora significativamente (1-3% su ImageNet).

Connessione con il bound PAC-Bayes

Per prior $P = \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ e posterior $Q = \mathcal{N}(\boldsymbol{\theta}, \sigma^2 \mathbf{I})$ (gaussiana centrata sul minimo), il bound PAC-Bayes:

$$R_T(Q) \leq \mathcal{L}_S(Q) + \sqrt{\frac{D_{\mathrm{KL}}(Q \| P) + \ln(n/\delta)}{2(n-1)}}$$

Con $D_{\mathrm{KL}}(Q\|P) = \|\boldsymbol{\theta}\|^2 / (2\sigma^2)$, minimizzare $\max_{\|\boldsymbol{\epsilon}\| \leq \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$ è equivalente a minimizzare il termine di generalizzazione del bound PAC-Bayes per una scelta appropriata di ρ e σ.

PARTE 91

Neural Tangent Kernel, Lazy Training e Reti Infinitamente Larghe

91.1 Derivazione del NTK

Per una rete $f_\theta : \mathbb{R}^d \to \mathbb{R}$ con parametri $\theta \in \mathbb{R}^p$, durante il training con gradient flow ($\dot{\theta} = -\nabla_\theta \mathcal{L}$), l'evoluzione dell'output su un input $\mathbf{x}$ è:

$$\dot{f}(\mathbf{x};\theta) = \nabla_\theta f(\mathbf{x};\theta) \cdot \dot{\theta} = -\nabla_\theta f(\mathbf{x};\theta) \cdot \nabla_\theta \mathcal{L}$$

Per la loss MSE $\mathcal{L} = \frac{1}{2N}\sum_{i=1}^{N}(f(\mathbf{x}_i;\theta) - y_i)^2$ con vettore di output $\mathbf{f} = (f(\mathbf{x}_1;\theta), \ldots, f(\mathbf{x}_N;\theta))^T$:

$$\dot{\mathbf{f}} = -\mathbf{K}^{(t)}(\mathbf{f} - \mathbf{y})/N$$

dove $\mathbf{K}^{(t)} \in \mathbb{R}^{N \times N}$ è il **Gram matrix del NTK**:

$$K^{(t)}(\mathbf{x}_i, \mathbf{x}_j) = \nabla_\theta f(\mathbf{x}_i;\theta_t)^T \nabla_\theta f(\mathbf{x}_j;\theta_t)$$

91.2 La Rete Infinitamente Larga e il Regime NTK

Jacot et al. (2018) dimostrano che per reti con larghezza $m \to \infty$ con la parametrizzazione NTK ($W_l \sim \mathcal{N}(0, 2/m^l)$), il kernel $K^{(t)}$ converge alla sua inizializzazione $K^{(0)} = K^\infty$ e vi rimane durante tutto il training.

In questo regime, la soluzione finale è:

$$\mathbf{f}(T) = \mathbf{y} + e^{-\mathbf{K}^\infty T/N}(\mathbf{f}(0) - \mathbf{y}) \xrightarrow{T\to\infty} \mathbf{y} + (e^{-\mathbf{K}^\infty \infty/N} - \mathbf{I})(\mathbf{f}(0) - \mathbf{y})$$

Con la rete inizializzata a zero ($\mathbf{f}(0) = \mathbf{0}$): $\mathbf{f}^\infty = (\mathbf{I} - e^{-\mathbf{K}^\infty/N})\mathbf{y} \to \mathbf{y}$ (convergenza al dataset). La predizione su un nuovo punto $\mathbf{x}^*$:

$$f^\infty(\mathbf{x}^*) = \mathbf{k}_*^T (\mathbf{K}^\infty)^{-1}\mathbf{y}$$

dove $k_*(\mathbf{x}_i) = K^\infty(\mathbf{x}^*, \mathbf{x}_i)$. Questa è *esattamente* la predizione del Kernel Regression con kernel K^∞ — le reti infinitamente larghe con gradient flow sono equivalenti ai kernel methods!

Il NTK per layer FC

Per una MLP con L layer nascosti e attivazione σ, il NTK si calcola ricorsivamente. Per un singolo layer: $K^{(1)}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$. Per layer profondi (senza bias):

$$K^{(h+1)}(\mathbf{x}, \mathbf{x}') = K^{(h)}(\mathbf{x}, \mathbf{x}')\dot{\Sigma}^{(h)}(\mathbf{x}, \mathbf{x}') + \Sigma^{(h+1)}(\mathbf{x}, \mathbf{x}')$$

$$\Sigma^{(h+1)}(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{f\sim\mathcal{GP}(0,\Sigma^{(h)})}[\sigma(f(\mathbf{x}))\sigma(f(\mathbf{x}'))]$$

$$\dot{\Sigma}^{(h)}(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{f\sim\mathcal{GP}(0,\Sigma^{(h)})}[\sigma'(f(\mathbf{x}))\sigma'(f(\mathbf{x}'))]$$

dove le aspettative si calcolano su reti gaussiane. Per ReLU: $\Sigma^{(h+1)} = \frac{1}{2\pi}(\sin\theta^{(h)} + (\pi - \theta^{(h)})\cos\theta^{(h)})\sqrt{\Sigma^{(h)}_{xx}\Sigma^{(h)}_{x'x'}}$ con $\cos\theta^{(h)} = \Sigma^{(h)}(\mathbf{x}, \mathbf{x}')/\sqrt{\Sigma^{(h)}(\mathbf{x}, \mathbf{x})\Sigma^{(h)}(\mathbf{x}', \mathbf{x}')}$.

PARTE 92

Memoria Associativa: Hopfield Classico e Moderno

92.1 Reti di Hopfield Classiche: Capacità e Dinamica

Una rete di Hopfield con N nodi memorizza p pattern $\{\boldsymbol{\xi}^\mu\}_{\mu=1}^{p} \in \{-1,+1\}^N$ tramite la regola di Hebb:

$$W_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu, \quad W_{ii} = 0$$

L'energia del sistema è $E(\boldsymbol{\sigma}) = -\frac{1}{2}\sum_{i\neq j} W_{ij}\sigma_i\sigma_j$. L'update asincrono $\sigma_i \leftarrow \text{sign}(\sum_j W_{ij}\sigma_j)$ discende l'energia.

Analisi della capacità tramite Signal-to-Noise

Per il recupero del pattern $\boldsymbol{\xi}^1$ dallo stato $\boldsymbol{\sigma} = \boldsymbol{\xi}^1$, il campo locale sul neurone i è:

$$h_i = \sum_j W_{ij}\xi_j^1 = \underbrace{\xi_i^1}_{\text{segnale}} + \underbrace{\frac{1}{N}\sum_{\mu=2}^{p}\sum_j \xi_i^\mu \xi_j^\mu \xi_j^1}_{\text{crosstalk noise}}$$

Il termine di crosstalk ha $p-1$ contributi, ognuno $\approx \mathcal{N}(0, 1/N)$ per pattern casuali. La varianza totale del rumore è $\approx (p-1)/N \approx p/N$. Per il recupero corretto: $\text{SNR} = 1/\sqrt{p/N} \gg 1$, quindi $p \ll N$. La **capacità critica** (Amit, Gutfreund, Sompolinsky, 1985) è $p_c \approx 0.138 N$.

92.2 Modern Hopfield Networks: Capacità Esponenziale

Ramsauer et al. (2021) propongono la generalizzazione con energia a interazione polinomiale di ordine n:

$$E(\boldsymbol{\xi}, \mathbf{X}) = -\text{lsexp}(\beta, \mathbf{X}^T\boldsymbol{\xi}) + \frac{1}{2}\|\boldsymbol{\xi}\|^2 + C$$

$$\text{lsexp}(\beta, \mathbf{z}) = \frac{1}{\beta}\log\sum_{\mu=1}^{M} e^{\beta z_\mu}$$

La regola di update che minimizza E:

$$\boldsymbol{\xi}^{\text{new}} = \mathbf{X}\,\text{softmax}(\beta\mathbf{X}^T\boldsymbol{\xi})$$

Con $\beta \to \infty$: $\boldsymbol{\xi}^{\text{new}} \to \mathbf{x}^{\mu^*}$ dove $\mu^* = \arg\max_\mu \mathbf{x}^{\mu T}\boldsymbol{\xi}$ — recupero esatto del pattern più simile.

Capacità: quanti pattern può memorizzare?

TEOREMA (RAMSAUER ET AL., 2021)

La Modern Hopfield Network può memorizzare e recuperare pattern casuali in $\mathbb{R}^d$ (normalizzati) senza errori con alta probabilità se:

$$M \leq \frac{1}{2\alpha^2}\exp\left(\frac{\alpha^2 d}{2\beta}\right)$$

dove $\alpha = 1/\beta\|\boldsymbol{\xi}\|$ è legato al rapporto segnale-rumore. La capacità cresce esponenzialmente in d — confronto con la capacità lineare $O(d)$ del Hopfield classico.

Connessione con l'Attention dei Transformer

Un singolo update della Modern Hopfield con query $\boldsymbol{\xi}$ e patterns $\mathbf{X} = \mathbf{K}$:

$$\boldsymbol{\xi}^{\text{new}} = \mathbf{K}\,\text{softmax}(\beta\mathbf{K}^T\boldsymbol{\xi}) \equiv \text{SoftmaxAttn}(\boldsymbol{\xi}, \mathbf{K}, \mathbf{K})$$

con $\beta = 1/\sqrt{d_k}$. Il meccanismo di attention è esattamente un passo di memoria associativa della Modern Hopfield Network — il Transformer implementa una memoria content-addressable con capacità esponenziale in ogni layer!

PARTE 93

Spectral Normalization e Stabilità delle GAN

93.1 Spectral Normalization

Per garantire che il discriminatore GAN sia 1-Lipschitz, la Spectral Normalization (Miyato et al., 2018) normalizza ogni layer per il suo massimo valore singolare. Per un layer FC con peso $\mathbf{W} \in \mathbb{R}^{m \times n}$:

$$\bar{\mathbf{W}} = \mathbf{W} / \sigma_1(\mathbf{W}), \qquad \sigma_1(\mathbf{W}) = \|\mathbf{W}\|_2 = \max_{\mathbf{h} \neq \mathbf{0}} \frac{\|\mathbf{W}\mathbf{h}\|_2}{\|\mathbf{h}\|_2}$$

Il massimo valore singolare si calcola efficacemente con **Power Iteration**: inizializzare $\tilde{\mathbf{v}}^0$ casualmente, poi iterare fino a convergenza:

$$\tilde{\mathbf{u}}^{l+1} = \mathbf{W}\tilde{\mathbf{v}}^l / \|\mathbf{W}\tilde{\mathbf{v}}^l\|, \quad \tilde{\mathbf{v}}^{l+1} = \mathbf{W}^T\tilde{\mathbf{u}}^{l+1} / \|\mathbf{W}^T\tilde{\mathbf{u}}^{l+1}\|$$
$$\sigma_1 \approx \tilde{\mathbf{u}}^{LT}\mathbf{W}\tilde{\mathbf{v}}^L$$

In pratica si usa una sola iterazione per step di training (aggiornando $\tilde{\mathbf{u}}, \tilde{\mathbf{v}}$ come variabili di stato) — la convergenza del power iteration a σ_1 è rapida se il gap tra primo e secondo valore singolare è grande.

Il gradiente di SN rispetto a W

Sia $\bar{\mathbf{W}} = \mathbf{W}/\sigma_1$. Il gradiente della loss rispetto a $\mathbf{W}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{1}{\sigma_1}\frac{\partial \mathcal{L}}{\partial \bar{\mathbf{W}}} - \frac{\partial \mathcal{L}}{\partial \bar{\mathbf{W}}} : \bar{\mathbf{W}} \cdot \frac{\tilde{\mathbf{u}}\tilde{\mathbf{v}}^T}{\sigma_1}$$

Il secondo termine sottrae la componente del gradiente nella direzione del vettore singolare dominante — questo è ciò che mantiene la norma spettrale uguale a 1 durante l'aggiornamento.

93.2 Progressioni e Trucchi per il Training delle GAN

Le GAN soffrono di training instabile per diverse ragioni:

- **Mode collapse:** il generatore collassa su un sottoinsieme dei modi della distribuzione reale — produce sempre le stesse immagini. Soluzione: Unrolled GAN (aggiorna il generatore rispetto al discriminatore futuro), Minibatch discrimination
- **Oscillazioni:** generatore e discriminatore si inseguono senza convergere. Soluzione: EMA dei pesi del generatore per l'inference (G_{ema})
- **Vanishing gradient:** il discriminatore diventa troppo bravo e il gradiente per il generatore è zero. Soluzione: WGAN-GP, hinge loss

La **hinge loss** (usata in SAGAN, BigGAN) è più stabile della BCE standard:

$$\mathcal{L}_D = \mathbb{E}[\max(0, 1 - D(\mathbf{x}_{\text{real}}))] + \mathbb{E}[\max(0, 1 + D(G(\mathbf{z})))]$$
$$\mathcal{L}_G = -\mathbb{E}[D(G(\mathbf{z}))]$$

PARTE 94

Teoria dell'Informazione Applicata al Deep Learning

94.1 Information Bottleneck e Compressione delle Rappresentazioni

Il principio dell'Information Bottleneck (Tishby e Schwartz-Ziv, 1999) cerca la rappresentazione T di X che massimizza l'informazione mutua con il target Y minimizzando quella con X:

$$\min_{p(t|x)} I(T;X) - \beta I(T;Y), \quad \beta > 0$$

La soluzione ottimale è definita da:

$$p(t|x) = \frac{p(t)}{Z(x,\beta)} \exp\left(-\beta \, D_{\mathrm{KL}}(p(y|x) \| p(y|t))\right)$$

dove $Z(x,\beta) = \sum_t p(t)\exp(-\beta D_{\mathrm{KL}}(\cdot))$ è la funzione di normalizzazione. Il parametro β controlla il trade-off: $\beta = 0$ minimizza $I(T;X)$ (massima compressione, nessuna informazione), $\beta \to \infty$ massimizza $I(T;Y)$ (nessuna compressione).

Information Plane nelle reti neurali

Tishby e Schwartz-Ziv (2017) proposero di visualizzare l'addestramento di ogni layer l come traiettoria nel piano $(I(T_l;X), I(T_l;Y))$ — il piano dell'informazione. La tesi: durante il training, c'è prima una fase di fitting (aumenta $I(T;Y)$) poi una fase di compressione (diminuisce $I(T;X)$). Questa interpretazione è controversa — la compressione dipende dalla metrica e dall'attivazione usata.

94.2 Minimum Description Length e Generalizzazione

Il principio MDL (Rissanen, 1978) lega la compressione alla generalizzazione. Il costo di codifica di un dataset $\mathcal{D}$ con un modello M:

$$L(M, \mathcal{D}) = L(M) + L(\mathcal{D}|M)$$

dove $L(M)$ è la lunghezza di descrizione del modello (bits per codificarlo) e $L(\mathcal{D}|M) = -\log_2 P_M(\mathcal{D})$ è il costo di codifica dei dati dato il modello. Il miglior modello minimizza la lunghezza totale — non solo la log-likelihood.

Variational MDL come ELBO

Con prior $P(\theta)$ e posterior $Q(\theta|\mathcal{D})$ sui parametri:

$$L_{\text{MDL}} = D_{\text{KL}}(Q \| P) - \mathbb{E}_Q[\log P(\mathcal{D}|\theta)]$$

Questo è l'ELBO negativo del VAE sui parametri — minimizzare l'ELBO sul modello è equivalente al principio MDL. Il termine D_{KL} penalizza la complessità del modello, il termine di likelihood premia il fit ai dati.

94.3 Information-Theoretic Bounds sulla Generalizzazione

Russo e Zou (2016) e Xu e Raginsky (2017) derivano bounds sulla generalizzazione usando la mutua informazione tra il training set S e l'ipotesi $W = A(S)$ prodotta dall'algoritmo A:

TEOREMA (RUSSO-ZOU / XU-RAGINSKY)

Per loss ℓ limitata in $[0, 1]$:

$$|\mathbb{E}[R(W)] - \mathbb{E}[\hat{R}_n(W)]| \leq \sqrt{\frac{I(W; S)}{2n}}$$

dove $I(W; S)$ è l'informazione mutua tra l'algoritmo output W e il dataset S.

Il bound lega la generalizzazione alla quantità di informazione che l'algoritmo "estrae" dal training set. Per SGD con rumore gaussiano di varianza σ^2,

$I(W;S) \leq \sum_{t=1}^{T} \frac{\alpha_t^2}{2\sigma_t^2} G_t^2$ — proporzionale al learning rate e inverso alla varianza del rumore.

PARTE 95

Federated Learning Avanzato: SCAFFOLD, Comunicazione Efficiente e Privacy

95.1 SCAFFOLD: Correzione del Client Drift

FedAvg soffre di client drift: ogni client ottimizza il proprio $\mathcal{L}_k$ che può essere molto diverso dal globale $\mathcal{L}$. SCAFFOLD (Karimireddy et al., 2020) introduce variabili di controllo $\mathbf{c}_k$ (stima del gradiente locale) e $\mathbf{c}$ (stima del gradiente globale) per correggere il drift:

$$\text{Update locale: } \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta(\nabla \mathcal{L}_k(\boldsymbol{\theta}) - \mathbf{c}_k + \mathbf{c})$$

$$\text{Aggiornamento controllo: } \mathbf{c}_k^+ = \mathbf{c}_k - \mathbf{c} + \frac{\boldsymbol{\theta}^0 - \boldsymbol{\theta}^E}{\eta E}$$

$$\text{Aggregazione server: } \mathbf{c} \leftarrow \mathbf{c} + \frac{1}{K}\sum_{k=1}^{K}(\mathbf{c}_k^+ - \mathbf{c}_k)$$

TEOREMA (CONVERGENZA SCAFFOLD)

Per funzioni L-smooth e μ-strongly convex, con K client, E step locali, learning rate η:

$$\mathbb{E}[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}^*\|^2] \leq \left(1 - \frac{\eta\mu}{2}\right)^T \cdot \frac{4\sigma^2}{\mu K}$$

Il bound non dipende dall'eterogeneità dei dati $\nabla\mathcal{L}_k(\boldsymbol{\theta}^*)$ — a differenza di FedAvg il cui bound peggiora proporzionalmente alla dissimilarità tra gradienti locali.

95.2 Compressione del Gradiente per la Comunicazione

La comunicazione è il collo di bottiglia nel federated learning. La compressione riduce i bit trasmessi con alcune tecniche principali:

TopK Sparsification: invia solo le k componenti con modulo maggiore:

$$\text{TopK}(\mathbf{g})_i = \begin{cases} g_i & |g_i| \in \text{top-}k \\ 0 & \text{altrimenti} \end{cases}$$

Con **error feedback**, l'errore di approssimazione viene accumulato e sommato al gradiente successivo: $\mathbf{e}_{t+1} = \mathbf{g}_t + \mathbf{e}_t - \text{TopK}(\mathbf{g}_t + \mathbf{e}_t)$. L'errore non si accumula indefinitamente: $\|\mathbf{e}_t\|^2 \leq (1 - k/d)\|\mathbf{g}_t\|^2$ se il gradiente non varia troppo.

Quantizzazione stocastica: arrotonda ogni componente a uno dei 2^b valori più vicini con probabilità proporzionale alla distanza:

$$Q_b(g_i) = \|g\|_2 \cdot \text{sign}(g_i) \cdot s_b\left(\frac{|g_i|}{\|g\|_2}\right)$$

dove s_b arrotonda stocasticamente al granello di quantizzazione più vicino su b bit. L'aspettativa è non polarizzata: $\mathbb{E}[Q_b(\mathbf{g})] = \mathbf{g}$.

95.3 Secure Aggregation

In molti scenari di federated learning, il server non deve vedere i gradienti individuali (privacy). Il protocollo di **Secure Aggregation** (Bonawitz et al., 2017) permette al server di calcolare $\sum_k \mathbf{g}_k$ senza vedere nessun $\mathbf{g}_k$ individuale usando maschere crittografiche:

$$\text{Client }k\text{ invia: }\mathbf{g}_k + \sum_{j>k}\mathbf{r}_{kj} - \sum_{j

dove $\mathbf{r}_{kj}$ sono maschere casuali condivise tra coppie di client (generate con protocollo Diffie-Hellman). Sommando tutti i messaggi ricevuti, le maschere si cancellano:

> $$\sum_k\!\left(\mathbf{g}_k + \sum_{j>k}\mathbf{r}_{kj} - \sum_{j
> Il server ottiene la somma ma non i contributi individuali. Il protocollo è sicuro anche se alcuni client si disconnettono durante il round, usando condivisione segreta (secret sharing) per gestire i dropout.

PARTE 96

Capsule Networks: Routing by Agreement e EM-Routing

96.1 Dynamic Routing by Agreement

Una **capsula** è un gruppo di neuroni il cui vettore di attività codifica parametri di istanziazione di un'entità (posizione, orientamento, deformazione) — la sua lunghezza rappresenta la probabilità di presenza. La **funzione di squashing** normalizza il vettore mantenendo la direzione:

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}\cdot\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}, \qquad \|\mathbf{v}_j\| \in [0,1)$$

dove $\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$ con $\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i$ la predizione della capsula bassa i per la capsula alta j.

Il routing by agreement aggiorna i coefficienti c_{ij} in base all'accordo tra $\hat{\mathbf{u}}_{j|i}$ e $\mathbf{v}_j$:

ALGORITMO — DYNAMIC ROUTING (R ITERAZIONI)

Inizializza: $b_{ij} = 0$ per tutti i, j. Per r iterazioni:

1. $c_{ij} = \text{softmax}_j(b_{ij}) = e^{b_{ij}} / \sum_k e^{b_{ik}}$
2. $\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$ (media pesata delle predizioni)
3. $\mathbf{v}_j = \text{squash}(\mathbf{s}_j)$
4. $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$ (accordo: aggiorna logit)

Il passo 4 è l'update fondamentale: se $\hat{\mathbf{u}}_{j|i}$ è nella stessa direzione di $\mathbf{v}_j$ (alto prodotto scalare), il routing coefficient c_{ij} aumenta — la capsula i contribuisce di più alla capsula j.

Proprietà: invarianza vs equivarianza

A differenza della max-pooling (che è invariante per piccole traslazioni), le capsule sono *equivarianti*: se l'input viene traslato, la lunghezza del vettore capsula (probabilità di presenza) rimane uguale, ma la sua direzione (parametri di istanziazione) cambia coerentemente con la traslazione.

96.2 EM-Routing: Formulazione come EM

Hinton et al. (2018) propongono EM-routing basato sulla mixture of Gaussians. Ogni capsula j al livello superiore è una componente gaussiana; le capsule i al livello inferiore sono i punti dati. Il routing è l'algoritmo EM:

E-step: aggiorna l'assegnamento (quale capsula superiore j spiega meglio la capsula inferiore i):

$$r_{ij} \propto \exp\left(\ln \alpha_j - \sum_{h} \frac{(v_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2} - \sum_h \ln \sigma_j^h \right)$$

dove v_{ij}^h è la h-esima coordinata del voto $\hat{\mathbf{u}}_{j|i}$, e α_j è il "activation" della capsula superiore.

M-step: aggiorna i parametri della distribuzione di ogni capsula superiore:

$$\mu_j^h = \frac{\sum_i r_{ij} v_{ij}^h}{\sum_i r_{ij}}, \quad (\sigma_j^h)^2 = \frac{\sum_i r_{ij} (v_{ij}^h - \mu_j^h)^2}{\sum_i r_{ij}}$$

L'activation α_j è determinata dal costo di codifica:
$\alpha_j = \sigma(-\lambda(\text{cost}_j^h - \beta_a))$ dove cost_j^h è il costo di descrivere i voti con la gaussiana j.

PARTE 97

Spiking Neural Networks: Analisi Completa

97.1 Modello LIF: Soluzione Analitica

Il modello Leaky Integrate-and-Fire (LIF) descrive il potenziale di membrana $u(t)$:

$$\tau_m \frac{du}{dt} = -(u - u_r) + R \cdot I(t)$$

dove $\tau_m = RC$ è la costante di membrana, u_r il potenziale di riposo, R la resistenza e $I(t)$ la corrente di input. La soluzione per input costante I:

$$u(t) = u_r + RI\left(1 - e^{-t/\tau_m}\right) + (u(0) - u_r)e^{-t/\tau_m}$$

Il tempo al primo spike t_{spike} si ottiene ponendo $u(t) = \vartheta$:

$$t_{\text{spike}} = \tau_m \ln\left(\frac{RI}{RI - (u(0) - u_r) - (\vartheta - u_r)}\right)$$

Solo se $RI > \vartheta - u_r$ (la corrente è sufficiente). Il rate di firing in regime stazionario (frequenza degli spike): $f = 1/t_{\text{ISI}}$ dove $t_{\text{ISI}} = \tau_m \ln(RI/(RI - (\vartheta - u_r)))$ è il tempo inter-spike.

97.2 Neural Coding: Rate vs Temporal

Le SNN trasmettono informazione tramite spike. Esistono due principali paradigmi di codifica:

Paradigma	Informazione in	Vantaggi	Applicazioni
Rate coding	Frequenza di firing in finestra temporale T: $f = N_{\text{spikes}}/T$	Robusto al rumore	Più computazionalmente trattabile
Temporal coding	Tempo esatto dei spike	Più efficiente in bits/spike	Compiti temporali precisi
Population coding	Pattern di attivazione su ensemble di neuroni	Ridondante e robusto	Corteccia sensoriale

97.3 Backpropagation Through Time per SNN

Per SNN con più step temporali T, si usa BPTT. La difficoltà: la funzione $s[t] = \mathbf{1}[u[t] \geq \vartheta]$ ha derivata zero quasi ovunque. Il **surrogate gradient** sostituisce la derivata con una funzione smooth:

$$\frac{\partial s}{\partial u}\bigg|_{\text{surrogate}} = \frac{1}{\pi}\frac{1}{1+(u-\vartheta)^2\pi^2} \quad \text{(fast sigmoid)}$$

$$\text{oppure: } \frac{\partial s}{\partial u}\bigg|_{\text{surrogate}} = \text{sign}(1-|u-\vartheta|)_+ \quad \text{(hat function)}$$

La BPTT su T step ha complessità $O(T)$ in memoria e tempo — costosa per T grandi (tipicamente $T = 4$–16). Il gradiente del potenziale rispetto ai pesi si propaga ricorsivamente:

$$\frac{\partial u[t]}{\partial w} = \frac{\partial u[t]}{\partial u[t-1]}\frac{\partial u[t-1]}{\partial w} + \frac{\partial u[t]}{\partial x[t]}\frac{\partial x[t]}{\partial w}$$

Il termine $\frac{\partial u[t]}{\partial u[t-1]} = \beta(1 - s[t-1])$ — quando c'è uno spike ($s = 1$), il potenziale viene resettato e il gradiente si azzera. Questo è l'analogo del "reset gate" nelle LSTM.

97.4 Spike-Timing Dependent Plasticity: Derivazione

La regola STDP modifica la forza sinaptica in base alla differenza temporale $\Delta t = t_{\text{post}} - t_{\text{pre}}$:

$$\Delta w = \begin{cases} A_+ e^{-\Delta t/\tau_+} & \Delta t > 0 \text{ (LTP: pre precede post)} \\ -A_- e^{\Delta t/\tau_-} & \Delta t < 0 \text{ (LTD: post precede pre)} \end{cases}$$

La STDP può essere derivata dalla regola di Hebb generalizzata. Consideriamo i **trace variables** x_{pre} e x_{post} che decadono esponenzialmente e vengono incrementati da ogni spike:

$$\dot{x}_{\text{pre}} = -x_{\text{pre}}/\tau_+ + \sum_t \delta(t - t_{\text{pre}}^{(t)})$$

$$\dot{x}_{\text{post}} = -x_{\text{post}}/\tau_- + \sum_t \delta(t - t_{\text{post}}^{(t)})$$

$$\dot{w} = A_+ x_{\text{pre}} \sum_t \delta(t - t_{\text{post}}^{(t)}) - A_- x_{\text{post}} \sum_t \delta(t - t_{\text{pre}}^{(t)})$$

Il primo termine incrementa w proporzionalmente a x_{pre} (quanto recente era l'ultimo spike pre) quando c'è uno spike post — implementa LTP. Il secondo decrementa w proporzionalmente a x_{post} quando c'è uno spike pre — implementa LTD.

PARTE 98

Quantum Machine Learning: VQC, Kernel Quantistici e Barren Plateaus

98.1 Variational Quantum Circuits

Un VQC è una sequenza di gate unitari con parametri $\boldsymbol{\theta} \in \mathbb{R}^p$. Lo stato quantistico iniziale $|0\rangle^{\otimes n}$ viene trasformato da un encoding layer $U_{\text{enc}}(\mathbf{x})$ (che dipende dall'input classico) e da layer variazionali $U(\boldsymbol{\theta})$:

$$|\psi(\mathbf{x}, \boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})\, U_{\text{enc}}(\mathbf{x})\, |0\rangle^{\otimes n}$$

La funzione obiettivo è il valore atteso di un osservabile $\hat{O}$ (tipicamente Pauli Z su un subset di qubit):

$$f(\mathbf{x}, \boldsymbol{\theta}) = \langle \psi(\mathbf{x}, \boldsymbol{\theta}) | \hat{O} | \psi(\mathbf{x}, \boldsymbol{\theta}) \rangle$$

Questa funzione è calcolabile su computer quantistico misurando $\hat{O}$ (con media su più shots per ridurre il rumore).

Parameter-Shift Rule: gradiente esatto

Per gate della forma $U(\theta_k) = e^{-i\theta_k H_k/2}$ con H_k un operatore hermitiano con autovalori ± 1 (tipicamente operatori di Pauli):

PARAMETER-SHIFT RULE

$$\frac{\partial f}{\partial \theta_k} = \frac{1}{2}\Big[f(\boldsymbol{\theta} + \frac{\pi}{2}\mathbf{e}_k) - f(\boldsymbol{\theta} - \frac{\pi}{2}\mathbf{e}_k) \Big]$$

Il gradiente esatto richiede 2 valutazioni del circuito per parametro — costo totale $O(p)$ per il gradiente completo.

Derivazione:
$f(\boldsymbol{\theta}) = \langle\psi|e^{i\theta_k H_k/2}\hat{O}e^{-i\theta_k H_k/2}|\psi\rangle = c_0 + c_1\cos\theta_k + c_2\sin\theta_k$.
Derivando: $\partial_{\theta_k} f = -c_1\sin\theta_k + c_2\cos\theta_k = \frac{f(\boldsymbol{\theta}+\pi/2) - f(\boldsymbol{\theta}-\pi/2)}{2}$.

98.2 Kernel Quantistici

Un **quantum kernel** usa il circuito di encoding per definire un kernel tra punti classici:

$$K_Q(\mathbf{x}, \mathbf{x}') = |\langle 0|^{\otimes n} U_{\text{enc}}^\dagger(\mathbf{x}') U_{\text{enc}}(\mathbf{x}) |0\rangle^{\otimes n}|^2 = |\langle\phi(\mathbf{x})|\phi(\mathbf{x}')\rangle|^2$$

dove $|\phi(\mathbf{x})\rangle = U_{\text{enc}}(\mathbf{x})|0\rangle^{\otimes n}$ è il feature map quantistico. Il kernel è computabile su hardware quantistico e rappresenta la sovrapposizione (similarità) degli stati quantistici. Schuld e Killoran (2022) mostrano che i VQC classificatori con output lineare sono equivalenti a SVM con kernel quantistico.

98.3 Barren Plateaus: Analisi Quantitativa

Il problema principale dei VQC: il gradiente scompare esponenzialmente con il numero di qubit.

TEOREMA (MCCLEAN ET AL., 2018 — BARREN PLATEAU)

Per VQC con layer t-design (circuiti che formano un ensemble t-design dal punto di vista della teoria delle misure), la varianza del gradiente è:

$$\text{Var}\left[\frac{\partial f}{\partial \theta_k}\right] = O\left(\frac{1}{2^n}\right)$$

Il gradiente è esponenzialmente piccolo nel numero di qubit n — per $n > 50$, il gradiente è numericamente indistinguibile da zero con precisione floating-point standard.

Intuizione: i circuiti quantistici profondi formano quasi un 2-design — distribuiscono lo stato uniformemente sulla sfera di Hilbert. Il volume

della sfera di Hilbert cresce esponenzialmente con n, quindi le variazioni di f sono esponenzialmente piccole rispetto alla scala del gradiente.

Soluzioni al barren plateau

- **Layerwise training:** addestrare un layer alla volta (come pre-training nei layer greedy)
- **Correlazione degli inizializzatori:** inizializzare parametri vicini in modo correlato
- **Circuiti locali:** usare solo osservabili locali (pochi qubit) — il barren plateau è meno severo
- **Separazione dei task:** usare un'architettura in cui layer vicini all'output non producono barren plateaus

PARTE 99

Optimal Control: Pontryagin, HJB e Connessioni con il Deep Learning

99.1 Il Principio del Massimo di Pontryagin

Il problema di controllo ottimale in tempo continuo cerca $\mathbf{u}:[0,T]\to\mathcal{U}$ che minimizza:

$$J[\mathbf{u}] = \int_0^T \ell(\mathbf{x}(t),\mathbf{u}(t))\,dt + \phi(\mathbf{x}(T))$$

$$\text{soggetto a: } \dot{\mathbf{x}}(t) = f(\mathbf{x}(t),\mathbf{u}(t)), \quad \mathbf{x}(0)=\mathbf{x}_0$$

Il **Principio del Massimo di Pontryagin (PMP)** fornisce le condizioni necessarie per l'ottimalità tramite l'Hamiltoniana del controllo:

$$\mathcal{H}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda},t) = \ell(\mathbf{x},\mathbf{u}) + \boldsymbol{\lambda}^T f(\mathbf{x},\mathbf{u})$$

PRINCIPIO DEL MASSIMO DI PONTRYAGIN

Se $(\mathbf{x}^*,\mathbf{u}^*)$ è ottimale, esiste la variabile aggiunta $\boldsymbol{\lambda}^*(t)$ tale che:

$$\dot{\mathbf{x}}^* = \frac{\partial \mathcal{H}}{\partial \boldsymbol{\lambda}} = f(\mathbf{x}^*,\mathbf{u}^*) \quad \text{(equazione di stato)}$$

$$\dot{\boldsymbol{\lambda}}^* = -\frac{\partial \mathcal{H}}{\partial \mathbf{x}} = -\nabla_{\mathbf{x}}\ell - (\nabla_{\mathbf{x}} f)^T\boldsymbol{\lambda}^* \quad \text{(equazione aggiunta)}$$

$$\mathbf{u}^*(t) = \arg\min_{\mathbf{u}\in\mathcal{U}} \mathcal{H}(\mathbf{x}^*(t),\mathbf{u},\boldsymbol{\lambda}^*(t),t) \quad \text{(condizione ottimalità)}$$

con condizione terminale $\boldsymbol{\lambda}^*(T) = \nabla_{\mathbf{x}}\phi(\mathbf{x}^*(T))$.

Connessione con le Neural ODE

Per una Neural ODE con parametri $\boldsymbol{\theta}$ e loss $\mathcal{L}(\mathbf{x}(T))$: lo stato è l'attivazione $\mathbf{x}(t)$, il "controllo" è la funzione f_θ (fissa durante l'inferenza, ottimizzata durante il training), e la variabile aggiunta $\boldsymbol{\lambda}(t) = \partial \mathcal{L}/\partial \mathbf{x}(t)$ è il gradiente. L'equazione aggiunta del PMP corrisponde alla backpropagation continua — il metodo adjoint è il PMP applicato al training.

99.2 Equazione di Hamilton-Jacobi-Bellman

La **funzione valore ottimale** $V(\mathbf{x}, t) = \min_{\mathbf{u}} \int_t^T \ell \, ds + \phi(\mathbf{x}(T))$ soddisfa la PDE di Hamilton-Jacobi-Bellman:

$$-\frac{\partial V}{\partial t} = \min_{\mathbf{u} \in \mathcal{U}} \left[\ell(\mathbf{x}, \mathbf{u}) + (\nabla_{\mathbf{x}} V)^T f(\mathbf{x}, \mathbf{u}) \right], \quad V(\mathbf{x}, T) = \phi(\mathbf{x})$$

La policy ottimale è $\mathbf{u}^*(\mathbf{x}, t) = \arg\min_{\mathbf{u}} \mathcal{H}(\mathbf{x}, \mathbf{u}, \nabla_{\mathbf{x}} V, t)$. La relazione con il PMP: $\boldsymbol{\lambda}^*(t) = \nabla_{\mathbf{x}} V(\mathbf{x}^*(t), t)$ — la variabile aggiunta è il gradiente della funzione valore.

Versione stocastica: HJB con rumore

Per sistemi stocastici $d\mathbf{x} = f(\mathbf{x}, \mathbf{u})\, dt + \sigma \, d\mathbf{w}$, l'HJB diventa:

$$-\frac{\partial V}{\partial t} = \min_{\mathbf{u}} \left[\ell + (\nabla_{\mathbf{x}} V)^T f + \frac{\sigma^2}{2} \operatorname{tr}(\nabla_{\mathbf{x}}^2 V) \right]$$

Il termine $\frac{\sigma^2}{2}\operatorname{tr}(\nabla_{\mathbf{x}}^2 V)$ è il laplaciano pesato della funzione valore — cattura l'effetto del rumore sulla propagazione del valore. Questa equazione è la base del RL stocastico e delle Neural ODE stocastiche (score-based models).

PARTE 100

Teoria del Campionamento, Aliasing e Rappresentazioni nelle Reti Neurali

100.1 Teorema di Nyquist-Shannon: Derivazione

Un segnale $x(t)$ a banda limitata con frequenza massima $f_{\max}$ (nessuna componente di Fourier sopra $f_{\max}$) può essere ricostruito esattamente dai campioni $x[n] = x(nT_s)$.

La prova usa la serie di Fourier dello spettro campionato. La Trasformata di Fourier del segnale campionato $x_s(t) = x(t) \cdot \text{III}_{T_s}(t) = \sum_n x(nT_s)\delta(t - nT_s)$:

$$X_s(f) = f_s \sum_{k} X(f - k f_s)$$

dove $f_s = 1/T_s$. Lo spettro campionato è la replica dello spettro originale ad intervalli di f_s. Se $f_s > 2f_{\max}$ (frequenza di Nyquist), le repliche non si sovrappongono e $X(f)$ si recupera con un filtro passa-basso ideale di banda $f_s/2$. Se $f_s < 2f_{\max}$: le repliche si sovrappongono → **aliasing** — frequenze alte vengono confuse con frequenze basse.

Formula di ricostruzione di Whittaker-Shannon

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT_s) \cdot \text{sinc}\left(\frac{t - nT_s}{T_s}\right), \quad \text{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$$

La funzione sinc è la risposta impulsiva del filtro passa-basso ideale — interpola esattamente tra i campioni per segnali a banda limitata.

100.2 Aliasing nelle CNN: Analisi Spettrale

Il pooling con stride s e il downsampling equivalgono al campionamento del segnale (feature map) con frequenza $f_{\text{s-input}}/s$. Se le feature map contengono componenti ad alta frequenza (sopra $f_{\text{s-input}}/(2s)$), si produce aliasing.

Zhang (2019) mostra che l'aliasing causa una mancanza di equivarianza per traslazione. Formalmente, sia T_δ la traslazione di δ pixel. Per una CNN con maxpool stride-2:

$$\| f(T_\delta \mathbf{x}) - T_\delta f(\mathbf{x}) \|_2 = O(\delta \cdot f_{\text{high}})$$

dove f_{high} è la potenza delle componenti ad alta frequenza nel feature map prima del downsampling. La soluzione: **BlurPool** applica un filtro anti-aliasing prima del campionamento:

$$\text{BlurPool}(\mathbf{x}) = (\mathbf{x} \star \mathbf{h})[::s]$$

dove $\mathbf{h}$ è un filtro gaussiano binomiale $\mathbf{h} = [1, 2, 1]/4$ (per stride 2) o $[1, 3, 3, 1]/8$ (Rect-3). Il filtro rimuove le componenti sopra $f_s/(2s)$ prima del downsampling.

100.3 Equivarianza e Aliasing: Misura Formale

La misura di equivarianza per traslazione di un modello f:

$$\text{Eq}(f) = 1 - \mathbb{E}_{\mathbf{x}, \delta}\left[\frac{\| f(T_\delta \mathbf{x}) - T_\delta f(\mathbf{x}) \|}{\| f(\mathbf{x}) \|} \right]$$

Zhang (2019) misura Eq su ResNet-50: dall'86% con maxpool standard al 99% con BlurPool — un miglioramento di 13 punti senza cambio dell'architettura principale. L'equivarianza migliora anche la robustezza alle corruzioni geometriche (rumore, ritagli, resize).

Analisi di Fourier delle CNN

Yin et al. (2020) analizzano le CNN nel dominio di Fourier. I modelli ResNet tendono ad avere alta sensibilità alle componenti ad alta frequenza (texture-bias): cambiare la texture di un'immagine (alte frequenze) a parità di forma (basse frequenze) cambia drasticamente la classificazione. Questo contrasta con la percezione umana (shape-bias). Il filtering delle alte frequenze durante il training (data augmentation con blur) riduce la texture-bias e migliora la robustezza alla distribuzione shift.

100.4 Spectral Analysis nell'Inizializzazione dei Pesi

La teoria del campionamento si connette all'inizializzazione tramite la **Spectral Analysis dell'Hessiana**. Al punto di inizializzazione, lo spettro dell'Hessiana $\mathbf{H}_0$ determina il comportamento iniziale del training:

$$\mathbf{H}_0 = \nabla^2 \mathcal{L}(\boldsymbol{\theta}_0) = \frac{1}{N}\mathbf{J}_0^T\mathbf{J}_0 + \frac{1}{N}\sum_i \mathbf{r}_i \nabla^2 \ell_i(\boldsymbol{\theta}_0)$$

dove $\mathbf{J}_0 = \nabla_\theta \mathbf{f}(\boldsymbol{\theta}_0)$ è la Jacobiana e $\mathbf{r}_i = f(\mathbf{x}_i;\boldsymbol{\theta}_0) - y_i$ è il residuo. Per inizializzazioni piccole ($\|\boldsymbol{\theta}_0\| \to 0$): $\mathbf{r}_i \to 0$ e $\mathbf{H}_0 \approx \frac{1}{N}\mathbf{J}_0^T\mathbf{J}_0$ — la matrice Gram del NTK, con autovalori positivi.

Il massimo autovalore $\lambda_{\max}(\mathbf{H}_0)$ determina il learning rate massimo sicuro: $\alpha_{\max} = 2/\lambda_{\max}$ (condizione di convergenza del GD). Per inizializzazione He su ReLU: $\lambda_{\max} \approx L/d$ dove L è la Lipschitz constant e d è la dimensione dell'input.

APPENDICE A

Glossario dei Simboli e delle Notazioni

Notazioni usate in tutto il Volume II, coerenti con il Volume I.

A.1 Insiemi, Spazi e Strutture Algebriche

Simbolo	Significato	Esempio d'uso
$\mathbb{R}^d$	Spazio euclideo reale d-dimensionale	$\mathbf{x} \in \mathbb{R}^d$ — vettore di input
$\mathbb{C}^d$	Spazio complesso d-dimensionale	Stato quantistico $\lvert\psi\rangle \in \mathbb{C}^{2^n}$
$\mathbb{Z}$, $\mathbb{Z}_n$	Interi; interi modulo n	Indici discreti di sequenze
$\mathbb{S}^2$	Sfera unitaria 2D in $\mathbb{R}^3$	Direzioni di visione in NeRF
Δ^{K-1}	Simplesso standard in $\mathbb{R}^K$: $\{\mathbf{p} \geq 0 : \sum_i p_i = 1\}$	Distribuzioni di probabilità discrete
$GL(\mathbb{R}^d)$	Gruppo delle mappe lineari invertibili su $\mathbb{R}^d$	Rappresentazioni di gruppi
$SO(n)$	Gruppo speciale ortogonale: rotazioni in $\mathbb{R}^n$	$R \in SO(3)$ — rotazione 3D
$SE(n)$	Gruppo speciale euclideo: rot. + trasl. in $\mathbb{R}^n$	Frame di proteine in AlphaFold
$\mathcal{H}$	Classe di ipotesi (insieme di funzioni)	VC dimension di $\mathcal{H}$
$\mathcal{F}$	σ-algebra (Teoria della Misura)	$(\Omega, \mathcal{F}, P)$ spazio di prob.
$\mathcal{B}(0, r)$	Palla euclide aperta di raggio r	Perturbazioni avversariali $\lVert\boldsymbol{\epsilon}\rVert \leq r$

A.2 Vettori e Matrici

Simbolo	Significato	Note
$\mathbf{x}, \mathbf{v}, \mathbf{w}$	Vettori colonna (minuscolo grassetto)	$\mathbf{x} \in \mathbb{R}^d$
$\mathbf{A}, \mathbf{W}, \mathbf{K}$	Matrici (maiuscolo grassetto)	$\mathbf{A} \in \mathbb{R}^{m \times n}$
$\mathbf{A}^T$	Trasposta di $\mathbf{A}$	$(\mathbf{A}^T)_{ij} = A_{ji}$
$\mathbf{A}^\dagger$, $\mathbf{A}^*$	Trasposta coniugata (hermitiana)	Matrici complesse: $A^\dagger_{ij} = \bar{A}_{ji}$
$\mathbf{A}^{-1}$	Inversa di $\mathbf{A}$ (se esiste)	$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$
$\|\mathbf{x}\|_2$, $\|\mathbf{x}\|$	Norma euclidea	$\sqrt{\sum_i x_i^2}$
$\|\mathbf{x}\|_1$	Norma L1	$\sum_i \|x_i\|$
$\|\mathbf{A}\|_F$	Norma di Frobenius	$\sqrt{\sum_{ij} A_{ij}^2} = \sqrt{\mathrm{tr}(\mathbf{A}^T\mathbf{A})}$
$\|\mathbf{A}\|_2 = \sigma_1(\mathbf{A})$	Norma spettrale (max valore singolare)	$\max_{\mathbf{x}\neq 0} \|\mathbf{A}\mathbf{x}\|/\|\mathbf{x}\|$
$\mathrm{tr}(\mathbf{A})$	Traccia: $\sum_i A_{ii}$	$\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$
$\det(\mathbf{A})$	Determinante	$\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$
$\mathbf{x} \otimes \mathbf{y}$	Prodotto tensoriale (outer product): $(\mathbf{x} \otimes \mathbf{y})_{ij} = x_i y_j$	Outer product mean in AlphaFold
$\mathbf{x} \odot \mathbf{y}$	Prodotto elemento per elemento (Hadamard)	Gating in LSTM, SNN
$\langle \mathbf{x}, \mathbf{y} \rangle$	Prodotto scalare $= \mathbf{x}^T\mathbf{y}$	Attenzione: $\langle \mathbf{q}, \mathbf{k} \rangle / \sqrt{d}$
$\|\psi\rangle$, $\langle\psi\|$	Ket (vettore colonna) e bra (vettore riga coniugato)	Notazione di Dirac per stati quantistici

A.3 Calcolo e Analisi

Simbolo	Significato	Note
$\nabla_{\mathbf{x}} f$	Gradiente di f rispetto a $\mathbf{x}$	Vettore $(\partial f/\partial x_i)_i$
$\nabla^2_{\mathbf{x}} f = \mathbf{H}$	Hessiana: matrice $(\partial^2 f/\partial x_i \partial x_j)_{ij}$	Curvatura locale
$\mathbf{J}_f(\mathbf{x})$	Jacobiana di $f:\mathbb{R}^d \to \mathbb{R}^m$: $(J_{ij}) = \partial f_i/\partial x_j$	$\mathbf{J} \in \mathbb{R}^{m\times d}$
$\partial f/\partial \mathbf{x}$	Derivata parziale (notazione compatta)	$= \nabla_{\mathbf{x}} f$ per scalari
f', $\dot{f}$	Derivata (punteggiata: rispetto al tempo)	$\dot{\mathbf{x}}(t) = d\mathbf{x}/dt$
$\mathrm{sg}[\cdot]$	Stop-gradient: identità nel forward, zero nel backward	VQ-VAE, IPO
$[x]_+ = \max(0, x)$	Parte positiva (rectifier)	ReLU, SVM loss
$\mathbf{1}[\cdot]$	Funzione indicatrice (1 se vera, 0 altrimenti)	$\mathbf{1}[u \geq \vartheta]$ negli spike
$\delta(t)$	Delta di Dirac	Spike neurali: $s(t) = \sum_k \delta(t - t_k)$
$\mathbb{E}[X]$	Valore atteso di X	$\int X\, dP$
$\mathrm{Var}[X]$	Varianza: $\mathbb{E}[(X - \mathbb{E}X)^2]$	Barren plateau: $\mathrm{Var}[\partial_\theta f]$
$P(A)$, $P(A\|B)$	Probabilità; probabilità condizionata	Bayes: $P(A\|B) = P(B\|A)P(A)/P(B)$
$\xrightarrow{P}$, $\xrightarrow{\text{q.c.}}$	Convergenza in probabilità; quasi	LGN: $\bar{X}_n \xrightarrow{P} \mu$

certamente

A.4 Notazione per Reti Neurali

Simbolo	Significato
$\boldsymbol{\theta}, \mathbf{w}$	Parametri della rete (da ottimizzare)
$f_\theta(\mathbf{x})$, $f(\mathbf{x};\boldsymbol{\theta})$	Output della rete con parametri $\boldsymbol{\theta}$ e input $\mathbf{x}$
$\mathbf{h}^{(\ell)}$, $\mathbf{z}^{(\ell)}$	Attivazioni (post-nonlinearità) e pre-attivazioni al layer ℓ
$\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}$	Pesi e bias del layer ℓ
$\sigma(\cdot)$	Funzione sigmoide $1/(1+e^{-x})$ oppure funzione di attivazione generica
$\phi(\cdot)$	Funzione di attivazione non-lineare (ReLU, GELU, SiLU...)
$\mathrm{softmax}(\mathbf{z})_i = e^{z_i}/\sum_j e^{z_j}$	Normalizzazione esponenziale; usata per probabilità e attenzione
$\mathrm{sparsemax}(\mathbf{z})$	Proiezione sul simplesso con soluzione sparsa (TabNet)
$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	Query, Key, Value nell'attenzione
$d_k, d_v, d_{\text{model}}$	Dimensioni di key/query, value, embedding
$\mathcal{L}(\boldsymbol{\theta})$	Loss function rispetto ai parametri $\boldsymbol{\theta}$
$R(h)$, $\hat{R}_N(h)$	Rischio vero; rischio empirico su N campioni
α, η	Learning rate
β (ottimizzatori)	Coefficiente di momentum
β (KL / diffusione)	Temperatura inversa; fattore di peso per KL
$\pi_\theta, \pi_{\text{ref}}$	Policy corrente e policy di riferimento (RLHF/DPO)
$\bar{\alpha}_t, \beta_t$	Schedule di rumore DDPM: $\bar{\alpha}_t = \prod_s (1-\beta_s)$

A.5 Notazione per Grafi

Simbolo	Significato
$G = (\mathcal{V}, \mathcal{E})$	Grafo con nodi $\mathcal{V}$ e archi $\mathcal{E}$
$N = \lvert\mathcal{V}\rvert$, $M = \lvert\mathcal{E}\rvert$	Numero di nodi e archi
$\mathbf{A} \in \{0,1\}^{N \times N}$	Matrice di adiacenza
$\mathbf{D} = \mathrm{diag}(d_1, \dots, d_N)$	Matrice dei gradi: $d_i = \sum_j A_{ij}$
$\mathbf{L} = \mathbf{D} - \mathbf{A}$	Laplaciano non normalizzato
$\mathbf{L}_{\mathrm{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$	Laplaciano normalizzato simmetrico
$\mathbf{h}_v^{(\ell)}$	Rappresentazione (embedding) del nodo v al layer ℓ
$\mathcal{N}(v)$	Insieme dei vicini del nodo v
$\mathrm{vol}(G) = \sum_i d_i$	Volume del grafo
$\lambda_1 \leq \lambda_2 \leq \cdots$	Autovalori del Laplaciano in ordine crescente

A.6 Notazione per Processi Stocastici e SDE

Simbolo	Significato
$\mathbf{w}(t)$, $d\mathbf{w}$	Moto Browniano standard; incremento Browniano
$p_t(\mathbf{x})$	Distribuzione marginale al tempo t
$\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$	Score function: gradiente del log-density
$\mathbf{s}_\theta(\mathbf{x}, t)$	Rete che stima lo score: $\mathbf{s}_\theta \approx \nabla \log p_t$
$p_{0t}(\mathbf{x}' \lvert \mathbf{x})$	Distribuzione condizionata: $\mathbf{x}'$ al tempo t dato $\mathbf{x}$ al tempo 0
$T_{\#}\mu$	Push-forward: distribuzione di $T(\mathbf{x})$ con $\mathbf{x} \sim \mu$
$\Pi(\mu, \nu)$	Insieme dei coupling con marginali μ e ν (Optimal Transport)
$W_p(\mu, \nu)$	Distanza di Wasserstein di ordine p
$\boldsymbol{\lambda}(t)$	Variabile aggiunta (costate) nell'Optimal Control

A.7 Funzioni e Operatori Speciali

Simbolo	Definizione	Proprietà chiave
$\mathrm{lsexp}(\mathbf{z}) = \log \sum_i e^{z_i}$	Log-sum-exp	$\max_i z_i \leq \mathrm{lsexp}(\mathbf{z})$
$D_{\mathrm{KL}}(P\|Q)$	Divergenza KL: $\mathbb{E}_P[\log(P/Q)]$	≥ 0; $= 0 \Leftrightarrow P = Q$
$I(X;Y)$	Informazione Mutua: $D_{\mathrm{KL}}(P_{XY}\|P_X \otimes P_Y)$	$I(X;Y) = H(X) -$
$H(X) = -\mathbb{E}[\log P(X)]$	Entropia di Shannon	$H \geq 0$; $H(\mathcal{U}_n) = \log$
$\mathrm{sinc}(t) = \sin(\pi t)/(\pi t)$	Funzione sinc	Filtro passa-basso ide Fourier
$\mathrm{III}_T(t) = \sum_n \delta(t - nT)$	Pettine di Dirac (Shah function)	Sampling: $x_s = x \cdot \mathrm{II}$
$\Phi(\cdot)$	CDF della gaussiana standard	Robustezza certificata
$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt$	Funzione Gamma	$\Gamma(n+1) = n!$; para

APPENDICE B

Identità Matematiche di Riferimento

B.1 Algebra Lineare

$$\det(e^{\mathbf{A}}) = e^{\operatorname{tr}(\mathbf{A})} \qquad (\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T \qquad (\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$$

Formula di identità matriciale di Woodbury:

$$(\mathbf{A}+\mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1}+\mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1}$$

Usata in: S4 (inversione della struttura NPLR), Gaussian Processes (inversione della covariance matrix con rank-1 update).

Disuguaglianza di von Neumann: Per matrici semidefinite positive $\mathbf{A}, \mathbf{B}$: $\operatorname{tr}(\mathbf{AB}) \leq \sum_i \sigma_i(\mathbf{A})\sigma_i(\mathbf{B})$.

Lemma di Schur: $\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{pmatrix} \succeq 0 \Leftrightarrow \mathbf{A} \succeq 0$ e $\mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} \succeq 0$.

B.2 Calcolo e Probabilità

Regola della catena: $\nabla_{\mathbf{x}}(f \circ g)(\mathbf{x}) = \mathbf{J}_g(\mathbf{x})^T \nabla f(g(\mathbf{x}))$.

Identità di Hutchinson: Per $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ o $\boldsymbol{\varepsilon} \sim \{\pm 1\}^d$ Rademacher: $\mathbb{E}[\boldsymbol{\varepsilon}^T\mathbf{A}\boldsymbol{\varepsilon}] = \operatorname{tr}(\mathbf{A})$.

$$\operatorname{Var}[\boldsymbol{\varepsilon}^T\mathbf{A}\boldsymbol{\varepsilon}] = 2\|\mathbf{A}\|_F^2 \quad \text{(gaussiana)}, \quad 2\sum_{i \neq j} A_{ij}^2 \quad \text{(Rademacher)}$$

Teorema di Bayes: $P(\boldsymbol{\theta}|\mathcal{D}) = P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})/P(\mathcal{D})$.

Disuguaglianza di Jensen: Per f convessa: $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$. Conseguenza: $D_{\mathrm{KL}} \geq 0$.

Diseguaglianza di Markov: $P(X \geq a) \leq \mathbb{E}[X]/a$ per $X \geq 0, a > 0$.

Disuguaglianza di Chebyshev: $P(|X - \mu| \geq k\sigma) \leq 1/k^2$.

Diseguaglianza di Hoeffding: Per $X_1, \ldots, X_n$ i.i.d. con $a_i \leq X_i \leq b_i$: $P(\bar{X} - \mathbb{E}[\bar{X}] \geq t) \leq \exp(-2n^2t^2 / \sum_i (b_i - a_i)^2)$. Usata nei bound di generalizzazione.

B.3 Identità per Funzioni di Attivazione

$$\sigma(x) = \frac{1}{1+e^{-x}}, \quad \sigma'(x) = \sigma(x)(1-\sigma(x))$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1, \quad \tanh'(x) = 1 - \tanh^2(x)$$

$$\text{softmax}(\mathbf{z})_i' = \text{softmax}(\mathbf{z})_i(\delta_{ij} - \text{softmax}(\mathbf{z})_j)$$

$$\text{GELU}(x) \approx x \cdot \sigma(1.702x), \quad \text{SiLU}(x) = x\sigma(x), \quad \text{Mish}(x) = x \tanh$$

B.4 Identità Utili per la Backpropagation

Per la forward pass $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$, i gradienti rispetto alla loss $\mathcal{L}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \mathbf{x}^T, \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \mathbf{W}^T \frac{\partial \mathcal{L}}{\partial \mathbf{y}}, \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}}$$

Per il prodotto di Hadamard $\mathbf{z} = \mathbf{a} \odot \mathbf{b}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \odot \mathbf{b}, \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \odot \mathbf{a}$$

Per l'attenzione scalare $a = \mathbf{q}^T\mathbf{k}/\sqrt{d}$ e output $\mathbf{o} = a\mathbf{v}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}} = \frac{1}{\sqrt{d}} \frac{\partial \mathcal{L}}{\partial a} \mathbf{k}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{k}} = \frac{1}{\sqrt{d}} \frac{\partial \mathcal{L}}{\partial a} \mathbf{q}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{v}} = a \frac{\partial \mathcal{L}}{\partial \mathbf{o}}$$

B.5 Identità per Processi Stocastici

Formula di Itô: Per X_t soluzione di SDE $dX_t = \mu_t\,dt + \sigma_t\,dW_t$ e funzione f due volte differenziabile:

$$df(X_t) = f'(X_t)\mu_t\,dt + f'(X_t)\sigma_t\,dW_t + \frac{1}{2}f''(X_t)\sigma_t^2\,dt$$

Il termine $\frac{1}{2}f''(\sigma_t^2)dt$ è la correzione di Itô — assente nel calcolo ordinario.

Lemma di Fokker-Planck (Liouville): La densità $p(\mathbf{x}, t)$ dell'ODE $\dot{\mathbf{x}} = f(\mathbf{x}, t)$ soddisfa: $\partial p/\partial t = -\nabla \cdot (fp)$, quindi $d\log p/dt = -\nabla \cdot f$.

APPENDICE C

Complessità Computazionale delle Operazioni Fondamentali

C.1 Operazioni di Base

Operazione	Complessità tempo	Complessità spazio	Note
Prodotto matrice-vettore $\mathbf{Ax}$ ($m \times n$)	$O(mn)$	$O(mn)$	Dominante nel forward pass
Prodotto matrice-matrice ($m \times k \times n$)	$O(mkn)$	$O(mk + kn)$	$O(n^3)$ per matrici quadrate
SVD di $\mathbf{A} \in \mathbb{R}^{m \times n}$	$O(\min(m,n)^2 \max(m,n))$	$O(mn)$	Spectral Norm
Inversione di $\mathbf{A} \in \mathbb{R}^{n \times n}$	$O(n^3)$	$O(n^2)$	LU o Cholesky
FFT su sequenza di lunghezza L	$O(L \log L)$	$O(L)$	S4, Hyena, convoluzione
Soft k-means / Sinkhorn (n punti, k cluster)	$O(nk)$ per iter.	$O(nk)$	OT-CFM, clustering spettrale

C.2 Complessità delle Architetture Principali

Modulo	Parametri	FLOPs (per token/elemento)	Memoria attivazioni
Attenzione Full (seq. L, dim. d)	$4d^2$	$O(L^2 d)$	$O(L^2)$ (score matrix)
Flash Attention (seq. L, dim. d)	$4d^2$	$O(L^2 d)$	$O(L)$ (tile-based)
Linear Attention	$4d^2$	$O(Ld^2)$	$O(d^2)$ (RNN state)
Feed-Forward (FFN) $d \to 4d \to d$	$8d^2$	$O(8Ld^2)$	$O(Ld)$
MoE (N esperti, top-k selezionati)	$N \cdot 8d^2$	$O(k \cdot 8Ld^2)$	$O(Ld)$
SSM/Mamba (seq. L, stato N, dim. d)	$O(dN + d^2)$	$O(LNd)$	$O(Nd)$ (RNN) o $O(L)$ (CNN)
GCN (grafo N nodi, M archi, dim. d)	d^2	$O((N+M)d^2)$	$O(Nd)$
Conv. 2D ($H \times W$, kernel k, canali C)	Ck^2	$O(HWC^2k^2)$	$O(HWC)$

C.3 Complessità degli Ottimizzatori

Ottimizzatore	Memoria extra (vs GD)	FLOPs extra per step	Note
SGD puro	0	0	Solo gradiente
SGD + Momentum	$\lvert\boldsymbol{\theta}\rvert$ (stato momentum)	$O(\lvert\boldsymbol{\theta}\rvert)$	Un vettore velocità
Adam / AdamW	$2\lvert\boldsymbol{\theta}\rvert$ (m, v)	$O(\lvert\boldsymbol{\theta}\rvert)$	Due vettori di stato
SAM	0	$2\times$ per step	Due forward+backward
LBFGS (m vettori history)	$2m\lvert\boldsymbol{\theta}\rvert$	$O(m\lvert\boldsymbol{\theta}\rvert)$	Approx. Newton efficiente

C.4 Scaling Laws: Stima del Budget Computazionale

Per un LLM Transformer con N parametri non-embedding e D token di training:

$$C \approx 6ND \quad \text{FLOPs} \qquad (\text{regola del pollice})$$

Derivazione: ogni token attraversa circa $2N$ operazioni (forward) e $4N$ (backward) — ma l'approssimazione $6ND$ è standard. Con le Chinchilla scaling laws ($D^* \approx 20N^*$):

$$C \approx 6N \cdot 20N = 120N^2 \quad \Rightarrow \quad N^*(C) \approx \sqrt{C/120}$$

Modello	Parametri	Token training	FLOPs totali (approx)
GPT-2 (Small)	117M	~40B	$\approx 2.8 \times 10^{19}$
GPT-3	175B	300B	$\approx 3.1 \times 10^{23}$
Chinchilla (70B ottimale)	70B	1.4T	$\approx 5.8 \times 10^{23}$
LLaMA-3 8B	8B	15T	$\approx 7.2 \times 10^{23}$
LLaMA-3 70B	70B	15T	$\approx 6.3 \times 10^{24}$

APPENDICE D

Iperparametri Consigliati e Diagnostica del Training

D.1 Tabella Iperparametri per Architettura

Architettura	Iperparametro	Valore tipico	Note
Transformer LLM	Learning rate	10^{-4}–3×10^{-4}	Con cosine decay
	Batch size (token)	2^{20}–2^{22}	Aumentare gradualmente
	Warmup steps	1–5% del training	Lineare da 0
	Weight decay (AdamW)	0.1	Su tutti tranne bias/norm
Vision Transformer	Patch size	16×16 o 32×32	Minore = più token
	Dropout	0.0–0.1	Spesso omesso con augm.
	Label smoothing	0.1	Regolarizzazione efficace
GNN	Layer	2–4	> 4: oversmoothing
	Hidden dim	64–256	Dipende da dimensione grafo
	Dropout sugli archi	0.5	DropEdge per regolariz.
Modelli Diffusione	Timestep T	1000 (DDPM)	50–100 con DDIM/Euler
	CFG weight w	7.5–12	Più alto = fedeltà↑, div↓
	Noise schedule	Cosine (DDPM)	Linear per immagini basse-res

RLHF / DPO	β (KL penalty)	0.01–0.5	Piccolo = più aggressivo
	Learning rate	10^{-5}–10^{-6}	Molto più basso del pretrain
	Batch size	32–128 coppie	Più grande = più stabile

D.2 Diagnostica: Sintomi e Cause

Sintomo	Cause probabili	Rimedi
Loss NaN all'inizio	Learning rate troppo alto; overflow numerico	Riduci LR; usa gradient clipping; mista precision
Loss scende poi si stabilizza alta	LR troppo basso; minimo locale; dati sbagliati	Aumenta LR; verifica dati; prova learning rate warmup
Training loss bassa, val. alta	Overfitting	Regularization (dropout, weight decay); più dati; data aug.
Loss oscilla molto	LR troppo alto; batch size troppo piccolo	Riduci LR; aumenta batch; gradient clipping
Gradienti vanishing	Rete troppo profonda; attivazioni saturate	Residual connections; batch norm; inizializzazione He/Xavier
Gradienti exploding	LR alto; inizializzazione sbagliata	Gradient clipping (≤ 1.0); normalizzazione; riduci LR
Training molto lento	Bottleneck CPU/I-O; micro-batch troppo piccolo	Data prefetching; aumenta batch; mixed precision (bf16)
GAN: mode collapse	Discriminatore troppo forte; LR sbilanciato	Riduci LR discriminatore; minibatch discrimination; WGAN-GP
GNN: oversmoothing	Troppi layer	Residual, JumpingKnowledge, GCNII; max 3–4 layer senza residual
LLM: repetizione	Temperatura bassa; presence/frequency penalty assenti	Aumenta temperatura; aggiungi repetition penalty

D.3 Checklist per il Debug del Training

CHECKLIST — VERIFICA SISTEMATICA

1. **Verifica il data pipeline:** controlla che la loss su un singolo batch scenda a zero (overfitting su un micro-dataset)
2. **Controlla l'inizializzazione:** la loss iniziale deve essere vicina al valore teorico atteso (es. $\log K$ per cross-entropy su K classi)
3. **Testa la backpropagation:** verifica il gradiente numericamente su un layer scelto casualmente
4. **Monitora la norma del gradiente:** deve essere $O(1)$ – se decresce a zero o sale, c'è un problema
5. **Verifica la learning rate:** la curva di LR range test deve mostrare un "dolce punto" prima dell'instabilità
6. **Controlla la normalizzazione:** le attivazioni di ogni layer devono avere media ≈ 0 e std ≈ 1 (BatchNorm/LayerNorm)
7. **Valuta la curva di apprendimento:** train loss deve scendere monotonicamente; se la val loss sale troppo presto, riduci la capacità del modello
8. **Usa EMA dei pesi:** la validation con $\mathrm{EMA}(\boldsymbol{\theta})$ è sempre migliore – utile per diagnosi

APPENDICE E

Bibliografia

Riferimenti organizzati per argomento, con focus sulle opere originali e sui survey più rilevanti trattati in questo volume.

E.1 Diffusione e Modelli Generativi Score-Based

- Ho, J., Jain, A., Abbeel, P. (2020). *Denoising Diffusion Probabilistic Models.* NeurIPS 2020. — DDPM: il modello di diffusione fondamentale con loss di denoising.
- Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B. (2021). *Score-Based Generative Modeling through Stochastic Differential Equations.* ICLR 2021. — Framework SDE unificato (VP-SDE, VE-SDE, Probability Flow ODE).
- Song, Y., Durkan, C., Murray, I., Ermon, S. (2021). *Maximum Likelihood Training of Score-Based Diffusion Models.* NeurIPS 2021. — Stima della likelihood tramite ODE.
- Ho, J., Salimans, T. (2022). *Classifier-Free Diffusion Guidance.* NeurIPS Workshop 2022. — Classifier-Free Guidance (CFG).
- Lipman, Y., Chen, R.T.Q., Ben-Hamu, H., Nickel, M., Le, M. (2022). *Flow Matching for Generative Modeling.* ICLR 2023. — Conditional Flow Matching con traiettorie lineari.
- Albergo, M.S., Vanden-Eijnden, E. (2023). *Building Normalizing Flows with Stochastic Interpolants.* ICLR 2023. — Stochastic Interpolants come generalizzazione di FM.

- Grathwohl, W., Chen, R.T.Q., Bettencourt, J., Sutskever, I., Duvenaud, D. (2019). *FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models.* ICLR 2019. — FFJORD con stimatore di Hutchinson.
- Kingma, D.P., Dhariwal, P. (2018). *Glow: Generative Flow with Invertible 1×1 Convolutions.* NeurIPS 2018. — Glow con decomposizione LU e ActNorm.
- Durkan, C., Bekasov, A., Murray, I., Papamakarios, G. (2019). *Neural Spline Flows.* NeurIPS 2019. — NSF con spline razionale-quadratiche.
- van den Oord, A., Vinyals, O., Kavukcuoglu, K. (2017). *Neural Discrete Representation Learning.* NeurIPS 2017. — VQ-VAE.
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B. (2022). *High-Resolution Image Synthesis with Latent Diffusion Models.* CVPR 2022. — Stable Diffusion.

E.2 Architetture per Sequenze e Memoria

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I. (2017). *Attention Is All You Need.* NeurIPS 2017. — Il Transformer originale.
- Gu, A., Goel, K., Ré, C. (2022). *Efficiently Modeling Long Sequences with Structured State Spaces.* ICLR 2022. — S4 con HiPPO e NPLR.
- Poli, M., Massaroli, S., Nguyen, E., Fu, D.Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., Ré, C. (2023). *Hyena Hierarchy: Towards Larger Convolutional Language Models.* ICML 2023. — Hyena con convoluzione implicita a lungo raggio.

- Zaheer, M., Guruganesh, G., Dubey, K.A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., Ahmed, A. (2020). *Big Bird: Transformers for Longer Sequences.* NeurIPS 2020. — BigBird e sua completezza di Turing.
- Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., et al. (2021). *Hopfield Networks is All You Need.* ICLR 2021. — Modern Hopfield Networks e connessione con Attention.
- Germain, M., Gregor, K., Murray, I., Larochelle, H. (2015). *MADE: Masked Autoencoder for Distribution Estimation.* ICML 2015. — MADE con mascheratura autoregressiva.
- van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., Kavukcuoglu, K. (2016). *Conditional Image Generation with PixelCNN Decoders.* NeurIPS 2016. — PixelCNN.
- Salimans, T., Karpathy, A., Chen, X., Kingma, D.P. (2017). *PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications.* ICLR 2017. — PixelCNN++.

E.3 Equivarianza, Grafi e Geometria

- Cohen, T.S., Welling, M. (2016). *Group Equivariant Convolutional Networks.* ICML 2016. — G-CNN con convoluzione su gruppo.
- Satorras, V.G., Hoogeboom, E., Welling, M. (2021). *E(n) Equivariant Graph Neural Networks.* ICML 2021. — EGNN equivariante per $E(3)$.
- Kipf, T.N., Welling, M. (2017). *Semi-Supervised Classification with Graph Convolutional Networks.* ICLR 2017. — GCN con

renormalization trick.

- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y. (2018). *Graph Attention Networks.* ICLR 2018. — GAT con attenzione su grafi.
- Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y. (2020). *Simple and Deep Graph Convolutional Networks.* ICML 2020. — GCNII con initial residual e identity mapping.
- Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M. (2018). *Modeling Relational Data with Graph Convolutional Networks.* ESWC 2018. — R-GCN per grafi eterogenei.
- Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J. (2018). *Hierarchical Graph Representation Learning with Differentiable Pooling.* NeurIPS 2018. — DiffPool.
- Von Luxburg, U. (2007). *A Tutorial on Spectral Clustering.* Statistics and Computing. — Survey completo su clustering spettrale.

E.4 Alignment, Preferenze e Sicurezza

- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., et al. (2022). *Training Language Models to Follow Instructions with Human Feedback.* NeurIPS 2022. — InstructGPT / RLHF.
- Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C.D., Finn, C. (2023). *Direct Preference Optimization: Your Language Model is Secretly a Reward Model.* NeurIPS 2023. — DPO.
- Azar, M.G., Rowland, M., Piot, B., Guo, D., Calandriello, D., Valko, M., Munos, R. (2024). *A General Theoretical Paradigm to*

Understand Learning from Human Feedback. AISTATS 2024. — IPO.

- Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., Kiela, D. (2024). *KTO: Model Alignment as Prospect Theoretic Optimization.* ICML 2024. — KTO ispirato alla Prospect Theory.
- Meng, Y., Xia, M., Chen, D. (2024). *SimPO: Simple Preference Optimization with a Reference-Free Reward.* NeurIPS 2024. — SimPO senza reference policy.
- Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., et al. (2022). *Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback.* arXiv:2204.05862. — Constitutional AI / Anthropic RLHF.
- Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N., Irving, G. (2022). *Red Teaming Language Models with Language Models.* arXiv:2202.03286. — Automated red teaming.

E.5 Ottimizzazione e Teoria dell'Apprendimento

- Nesterov, Y. (1983). *A Method for Solving the Convex Programming Problem with Convergence Rate* $O(1/k^2)$. Soviet Mathematics Doklady. — NAG con ottimalità $O(1/T^2)$.
- Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B. (2021). *Sharpness-Aware Minimization for Efficiently Improving Generalization.* ICLR 2021. — SAM.
- Jacot, A., Gabriel, F., Hongler, C. (2018). *Neural Tangent Kernel: Convergence and Generalization in Neural Networks.* NeurIPS 2018.

— NTK e regime lazy training.

- Jin, C., Ge, R., Netrapalli, P., Kakade, S.M., Jordan, M.I. (2017). *How to Escape Saddle Points Efficiently.* ICML 2017. — Perturbed GD e convergenza ai SOSP.
- Belkin, M., Hsu, D., Ma, S., Mandal, S. (2019). *Reconciling Modern Machine Learning Practice and the Classical Bias-Variance Trade-Off.* PNAS 2019. — Double descent.
- Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W. (2010). *A Theory of Learning from Different Domains.* Machine Learning. — Bound di generalizzazione con domain shift.
- Vovk, V., Gammerman, A., Shafer, G. (2005). *Algorithmic Learning in a Random World.* Springer. — Conformal Prediction.
- Russo, D., Zou, J. (2016). *Controlling Bias in Adaptive Data Analysis Using Information Theory.* COLT 2016. — Bound MI sulla generalizzazione.
- Pilanci, M., Ergen, T. (2023). *Neural Networks are Convex Regularizers: Exact Polynomial-Time Convex Optimization Formulations for Two-Layer Networks.* ICML 2020. — Convexification per reti a 2 layer.
- Duchi, J., Hazan, E., Singer, Y. (2011). *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.* JMLR 2011. — AdaGrad.
- Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S.J., Stich, S.U., Suresh, A.T. (2020). *SCAFFOLD: Stochastic Controlled Averaging for Federated Learning.* ICML 2020. — SCAFFOLD.

E.6 Architetture Avanzate e Modelli Fondazionali

- Hoffmann, J., Borgeaud, S., Mensch, A., et al. (2022). *Training Compute-Optimal Large Language Models.* NeurIPS 2022. — Chinchilla scaling laws.
- Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. (2022). *Emergent Abilities of Large Language Models.* TMLR 2022. — Emergenza nei LLM.
- Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., et al. (2021). *Highly Accurate Protein Structure Prediction with AlphaFold.* Nature. — AlphaFold2 completo.
- Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R. (2020). *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.* ECCV 2020. — NeRF.
- Sitzmann, V., Martel, J.N.P., Bergman, A.W., Lindell, D.B., Wetzstein, G. (2020). *Implicit Neural Representations with Periodic Activation Functions.* NeurIPS 2020. — SIREN.
- Müller, T., Evans, A., Schied, C., Keller, A. (2022). *Instant Neural Graphics Primitives with a Multiresolution Hash Encoding.* ACM SIGGRAPH 2022. — Instant-NGP.
- Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I. (2023). *Robust Speech Recognition via Large-Scale Weak Supervision.* ICML 2023. — Whisper.
- Gu, A., Dao, T. (2024). *Mamba: Linear-Time Sequence Modeling with Selective State Spaces.* arXiv:2312.00752. — Mamba SSM.

- Liu, H., Li, C., Wu, Q., Lee, Y.J. (2023). *Visual Instruction Tuning.* NeurIPS 2023. — LLaVA.
- Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). *Learning Transferable Visual Models From Natural Language Supervision.* ICML 2021. — CLIP.

E.7 Biologia Computazionale, Fisica e Frontiere

- Cohen, T.S., Welling, M. (2016). *Steerable CNNs.* ICLR 2017. — CNN con filtri steerables e teoria delle rappresentazioni.
- Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y. (2018). *Spectral Normalization for Generative Adversarial Networks.* ICLR 2018. — SN-GAN.
- Zhang, R. (2019). *Making Convolutional Networks Shift-Invariant Again.* ICML 2019. — BlurPool e aliasing nelle CNN.
- Hopfield, J.J. (1982). *Neural Networks and Physical Systems with Emergent Collective Computational Abilities.* PNAS 1982. — Rete di Hopfield classica.
- Maas, A.L., Hannun, A.Y., Ng, A.Y. (2013). *Rectifier Nonlinearities Improve Neural Network Acoustic Models.* ICML 2013. — Leaky ReLU.
- Schuld, M., Killoran, N. (2022). *Quantum Machine Learning in Feature Hilbert Spaces.* Physical Review Letters. — Quantum kernels.
- McClean, J.R., Boixo, S., Smelyanskiy, V.N., Babbush, R., Neven, H. (2018). *Barren Plateaus in Quantum Neural Network Training*

Landscapes. Nature Communications. — Barren plateaus.

- Maaten, L.v.d., Hinton, G. (2008). *Visualizing Data using t-SNE.* JMLR 2008. — t-SNE (citato nelle diagnostiche).
- Arjovsky, M., Chintala, S., Bottou, L. (2017). *Wasserstein GAN.* ICML 2017. — WGAN con W_1-distance.
- Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A. (2017). *Improved Training of Wasserstein GANs.* NeurIPS 2017. — WGAN-GP con gradient penalty.
- Kahneman, D., Tversky, A. (1979). *Prospect Theory: An Analysis of Decision under Risk.* Econometrica. — Fondamento teorico di KTO.
- Hinton, G.E., Sabour, S., Frosst, N. (2018). *Matrix Capsules with EM Routing.* ICLR 2018. — EM Routing nelle Capsule Networks.
- Mahowald, K., Ivanova, A.A., Blank, I.A., Kanwisher, N., Tenenbaum, J.B., Fedorenko, E. (2024). *Dissociating Language and Thought in Large Language Models: A Cognitive Perspective.* Trends in Cognitive Sciences. — Riflessioni cognitive sui LLM.

E.8 Libri di Testo e Survey Fondamentali

- Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning.* MIT Press. — Il riferimento fondamentale per il deep learning.
- Bishop, C.M. (2006). *Pattern Recognition and Machine Learning.* Springer. — Probabilistic perspective sull'apprendimento automatico.
- Murphy, K.P. (2022). *Probabilistic Machine Learning: An Introduction / Advanced Topics.* MIT Press (2 voll.). — Trattazione moderna e probabilistica.

- Shalev-Shwartz, S., Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press. — Teoria dell'apprendimento (PAC, VC).
- Boyd, S., Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press. — Ottimizzazione convessa e dualità.
- Cover, T.M., Thomas, J.A. (2006). *Elements of Information Theory.* Wiley. — Teoria dell'informazione rigorosa.
- Hamilton, W.L. (2020). *Graph Representation Learning.* Synthesis Lectures on AI and Machine Learning. — Survey completo sulle GNN.
- Bommasani, R., Hudson, D.A., Aditi, E., et al. (2021). *On the Opportunities and Risks of Foundation Models.* arXiv:2108.07258. — Survey fondamentale sui foundation models.

Matematica delle Reti Neurali — Volume II · Silvano Salvador

Parti 51–95 · Continuazione del Volume I (Parti 0–50)

Parti 51–100 · Appendici A–E · Glossario simboli · Identità matematiche · Complessità computazionale · Diagnostica training · Bibliografia (80+ riferimenti) · PINN · Score Models · Consistency · BPE · Word2Vec · Beam Search · Speculative Decoding · Long Context · DeepSeek MLA · MCTS · Reward Shaping · Multi-Task · Causalità · TDA · Varietà Riemanniane · SE(3) · FNO · MoD · Active Learning